$N°$ d'Ordre : D.U. xxxx
EDSPIC : xxx

Université Paris 13 - Insitut Galilée

Laboratoire d'Informatique de Paris Nord
UMR 7030 du CNRS

Thèse

présentée pour l'obtention du titre de
Docteur en Sciences
spécialité
Informatique
par

Nicoleta ROGOVSCHI

Classification à base de modèles de mélanges topologiques des données catégorielles et continues

---

Soutenue publiquement le 4 décembre 2009 devant le jury :

Younès BENNANI	Professeur Université Paris 13	Directeur de thèse
Mustapha LEBBAH	Maître de conférences Université Paris 13	Co-encadrant
Djamel BOUCHAFFRA	Professeur Grambling State University	Rapporteur
Mohamed NADIF	Professeur Université Paris 5	Rapporteur
Frederic ALEXANDRE	Directeur de recherche INRIA Nancy	Examinateur
Khalid BENABDESLEM	Maître de conférences Université Lyon 1	Examinateur
Bruce DENBY	Professeur Université Paris 6	Examinateur
Catherine RECANATI	Maître de conférences Université Paris 13	Examinateur

Résumé

Le travail de recherche exposé dans cette thèse concerne le développement d'approches à base de cartes auto-organisatrices dans un formalisme de modèles de mélanges pour le traitement de données qualitatives, mixtes et séquentielles. Pour chaque type de données, un modèle d'apprentissage non supervisé adapté est proposé. Le premier modèle, décrit dans cette étude, est un nouvel algorithme d'apprentissage des cartes topologiques BeSOM (Bernoulli Self-Organizing Map) dédié aux données binaires. Chaque cellule de la carte est associée à une distribution de Bernoulli. L'apprentissage dans ce modèle a pour objectif d'estimer la fonction densité sous forme d'un mélange de densités élémentaires. Chaque densité élémentaire est-elle aussi un mélange de lois Bernoulli définies sur un voisinage. Le second modèle aborde le problème des approches probabilistes pour le partitionnement des données mixtes (quantitatives et qualitatives). Le modèle s'inspire de travaux antérieurs qui modélisent une distribution par un mélange de lois de Bernoulli et de lois Gaussiennnes. Cette approche donne une autre dimension aux cartes topologiques : elle permet une interprétation probabiliste des cartes et offre la possibilité de tirer profit de la distribution locale associée aux variables continues et catégorielles. En ce qui concerne le troisième modèle présenté dans cette thèse, il décrit un nouveau formalisme de mélanges Markovien dédiés au traitement de données structurées en séquences. L'approche que nous proposons est une généralisation des chaines de Markov traditionnelles. Deux variantes sont développées : une approche globale où la topologie est utilisée d'une manière implicite et une approche locale où la topologie est utilisée d'une manière explicite. Les résultats obtenus sur la validation des approches traités dans cette étude sont encourageants et prometteurs à la fois pour la classification et pour la modélisation.

Mots clés : apprentissage non-supérvisé, cartes auto-organisatrices, modèles de mélanges, chaînes de Markov, données catégorielles, données mixtes, données séquentielles.

Abstract

The research presented in this thesis concerns the development of self-organising map approaches based on mixture models which deal with different kinds of data : qualitative, mixed and sequential. For each type of data we propose an adapted unsupervised learning model. The first model, described in this work, is a new learning algorithm of topological map BeSOM (Bernoulli Self-Organizing Map) dedicated to binary data. Each map cell is associated with a Bernoulli distribution. In this model, the learning has the objective to estimate the density function presented as a mixture of densities. Each density is as well a mixture of Bernoulli distribution defined on a neighbourhood. The second model touches upon the problem of probability approaches for the mixed data clustering (quantitative and qualitative). The model is inspired by previous works which define a distribution by a mixture of Bernoulli and Gaussian distributions. This approach gives a different dimension to topological map : it allows probability map interpretation and offers the possibility to take advantage of local distribution associated with continuous and categorical variables. As for the third model presented in this thesis, it is a new Markov mixture model applied to treatment of the data structured in sequences. The approach that we propose is a generalisation of traditional Markov chains. There are two versions : the global approach, where topology is used implicitly, and the local approach where topology is used explicitly. The results obtained upon the validation of all the methods are encouraging and promising, both for classification and modelling.

Key words : unsupervised learning, self-organised-maps, mixture models, hidden Markov models, mixed data, sequential data, categorical data.

Table des matières

Table des figures

Liste des tableaux

Notations

$\mathcal{A}$: ensemble d'apprentissage, $\mathcal{A} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$
N : nombre d'exemples de la base d'apprentissage $\mathcal{A}$
$\mathbf{x}_i$: un élément de $\mathcal{A}$
$\mathbf{z}$: variable cachée
$\mathcal{C}$: carte topologique
c : cellule de la carte $\mathcal{C}$
K : nombre de cellules de la carte
$\mathbf{w}_c$: référent relatif à la cellule c
$\mathbf{f}_c$: fonction densité gaussienne relative à la cellule c
$\mathcal{H}$: distance de Hamming
t : indice d'étape d'apprentissage, itération
T : temperature
T_{min} : temperature initiale
T_{max} : temperature finale
N_{iter} : nombre d'itérations
V : vraisemblance
I : matrice identité
σ_c : écart-type
ε : probabilité associée à la loi de Bernoulli
$\delta(.,.)$: distance entre deux cellules de la carte
$\mathcal{K}(.)$: fonction de voisinage

Introduction

Ce travail de thèse a été réalisé dans l'équipe A3 (Apprentissage Artificiel & Applications) du laboratoire LIPN (Laboratoire d'Informatique de Paris-Nord), UMR 7030. L'objectif général de mon travail était de développer des modèles à base de modèles de mélanges pour l'analyse de données qualitatives et mixtes. Ce travail se situe dans le domaine de l'informatique et de l'analyse de données en utilisant des techniques d'apprentissage non supervisé à base de modèles de mélanges.

L'apprentissage non supervisé consiste à construire des représentations simplifiées de données, pour mettre en évidence les relations existantes entre les caractéristiques relevées sur des données et les ressemblances ou dissemblances de ces dernières, sans avoir aucune connaissance sur les classes. On peut distinguer deux grandes familles : les méthodes probabilistes et les méthodes déterministes ou tout simplement les méthodes de quantification. Ce travail concerne le traitement des données qualitatives, mixtes et les données structurées en séquence à l'aide des cartes auto-organisatrices, [98, 5, 149] dans le cadre du formalisme des modèles de mélanges.

Les cartes topologiques utilisent un algorithme d'auto-organisation (Self-Organizing Map, SOM) qui permet d'une part de quantifier de grandes quantités de données en regroupant les observations similaires en groupes ou clusters et d'autre part de projeter les groupes obtenus de façon non-linéaire sur une carte, permettant ainsi de visualiser la structure du jeu de données en deux dimensions, tout en respectant la topologie initiale des données. La mise en œuvre de l'algorithme sur des données qualitatives et mixtes suppose toujours une phase de prétraitement permettant d'extraire une information numérique des observations pour la partie qualitative. Des travaux antérieurs [105, 103] ont permis de proposer un modèle de cartes topologiques déterministes pour les données binaires et un autre modèle probabiliste dédié au données catégorielles.

Différentes approches ont été envisagées, pour différents types de données, reposant

sur des formalismes probabilistes. Dans [150], les auteurs proposent une généralisation probabiliste des cartes auto-organisatrices qui maximise l'énergie variationnelle et qui additionne la log-vraisemblance des données et la divergence de Kullback-Leibler entre une fonction de voisinage normalisée et la distribution postérieure sur les données pour les composants. Nous avons également STVQ (Soft topographic vector quantization), qui emploie une certaine mesure de divergence entre les données élémentaires et les référents afin de minimiser une nouvelle fonction d'erreur [83, 74]. Il existe aussi un modèle original qui permet de créer un graphe entre les prototypes en se basant sur les modèles de mélanges et les graphes de Delaunay [9]. Un autre modèle, souvent présenté comme la version probabiliste des cartes auto-organisatrices, est GTM (Generative Topographic Map) [22, 91]. Cependant, la façon dont GTM atteint l'organisation topologique est très différente de celle utilisée dans les modèles des cartes topologiques traditionnelles. Dans GTM le mélange est paramétré par une combinaison linéaire de fonctions non linéaires des positions des cellules de la carte (GTM a été conçu pour les données quantitatives). Des extensions de ce modèle à un modèle dédié aux données discrètes et binaires ont été proposées [72, 132, 133].

La difficulté principale en ce qui concerne l'application des modèles de mélanges de distributions multivariables pour le partitionnement "clustering" est qu'ils sont liés à un type de données : les distributions normales pour les données quantitatives, la distribution de Bernoulli pour les données binaires, et le modèle "Latent class" pour les variables catégorielles. Ceci est limitatif puisque la plupart des ensembles de données sont mixtes et impliquent différents types d'informations : quantitatifs, catégoriels, binaires et séquentielles.

L'idée de base de ce travail repose sur le principe de la conservation de la structure initiale des données en utilisant le formalisme probabiliste. Les modèles de mélanges proposés ici vérifient ce principe et fournissent des résultats directement interprétables par rapport aux données initiales, qu'elles soient simplement binaires, mixtes ou structurées en séquences.

Pour comprendre toutes les étapes qui ont mené à l'élaboration de cette thèse, le plan suivant a été retenu :

Le premier chapitre est consacré à la présentation des modèles d'apprentissage non supervisé et principalement les méthodes de partitionnement. Il présente le contexte de l'étude de la classification sur de grandes bases de données, en expliquant en détail différentes approches de classification non supervisée. Le partitionnement "clustering" peut être défini comme la division des données en groupes/clusters de données similaires. L'apprentissage de grandes bases de données décrites à l'aide de variables différentes imposent aux algorithmes de classification à base d'apprentissage des contraintes liées aux types de variables, ou à la distribution associée...etc. Ces difficultés ont conduit à l'émergence d'assez puissants modèles de classification. Ce chapitre n'a pas la prétention de l'exhaustivité, nous nous sommes limités à proposer une vue ou une représentation assez synthétique des méthodes de partitionnement "clustering" qui existent dans différentes catégories (c'est-à-dire hiérarchique, par partitionnement, à base de densité, à base de grille). Dans le cadre de cette thèse, nous nous sommes intéressés plus particulièrement aux algorithmes à base de modèles de mélanges et des cartes auto-organisatrices. Un autre problème important que nous avons discuté dans le cadre de ce chapitre est l'évaluation des résultats de la classification.

Le chapitre 2 présente à un nouveau modèle probabiliste proposé dans cette thèse dédié aux données binaires "BeSOM" (Bernoulli Self-Organizing Map). Après un bref rappel du modèle de base appliqué aux données continues PrSOM [5] et la loi de Bernoulli utilisée dans les travaux [125, 35], nous présentons le modèle BeSOM suivi de son algorithme d'apprentissage. En effet, dans le but de donner une interprétation probabiliste des cartes topologiques, nous avons développé un modèle probabiliste des cartes topologiques dédiées aux données binaires. Chaque cellule de la carte est associée à une distribution de Bernoulli. L'apprentissage dans ce modèle a pour objectif d'estimer la fonction densité sous forme d'un mélange de densités élémentaires. Chaque densité élémentaire est elle aussi un mélange de lois Bernoulli définies sur un voisinage. Ce nouvel algorithme d'apprentissage probabiliste et non supervisé utilise l'algorithme EM pour maximiser la vraisemblance des données afin d'estimer les paramètres du modèle de mélange de lois de Bernoulli. Nous montrons sa convergence et nous développons toutes les formules nécessaires. L'algorithme a une portée pratique, nous montrons sur des exemples l'amélioration qu'il apporte aussi bien en classification qu'en visualisation.

Le chapitre 3 aborde le problème des modèles probabilistes dédiés aux données mixtes (continues et qualitatives). Nous montrons dans ce chapitre comment il est

possible de modifier et d'étendre le modèle BeSOM afin d'en extraire des informations probabilistes pour la classification non supervisée des données mixtes. Le modèle que nous développons dans ce chapitre est une extension du modèle PrSOM qui est l'un des premiers modèles probabilistes dédiés aux données continues. Pour définir notre modèle de mélanges manipulant des données mixtes, nous avons modélisé la distribution par un mélange de mélange de lois de Bernoulli et de lois Gaussiennes. Cet algorithme d'apprentissage non supervisé PrMTM (Probabilistic Mixed Topologial Map) est une application de l'algorithme EM pour maximiser la vraisemblance. L'algorithme PrMTM donne une autre dimension aux cartes topologiques : il permet une interprétation probabiliste des cartes et il permet de tirer profit de la distribution locale associée aux variables continues et catégorielles. Cet algorithme a l'avantage de fournir des prototypes qui sont de la même nature que les données initiales. L'algorithme PrSOM et BeSOM apparaissent alors comme un cas particulier de l'algorithme PrMTM.

Le chapitre 4 présente l'extension des modèles probabilistes proposés dans cette thèse aux données structurées en séquences. En effet, depuis plusieurs années, les séquences temporelles et spatiales ont été le sujet de recherche dans plusieurs domaines tels que : la statistique, la reconnaissance des formes, et la bioinformatique. La méthode la plus facile pour traiter les données structurées en séquence serait tout simplement d'ignorer l'aspect séquentiel et de traiter les observations comme des données indépendantes ou i.i.d "independent and identically distributed". Pour beaucoup d'applications, l'hypothèse i.i.d rend les données plus pauvres en perdant l'information séquentielle. Dans ce chapitre nous développons une extension d'un modèle naturel pour le traitement de données séquentielles qui est le modèle des chaînes de Markov cachées (HMM). Deux approches sont décrites pour construire un modèle de mélange markovien topologique. Concernant la première approche, nous fournissons le détail des expressions permettant d'estimer les paramètres du modèle en utilisant l'algorithme EM avec l'algorithme Baum-Welch qui intègrent la topologie du graphe associé au modèle HMM. La deuxième approche développée dans ce chapitre est proche des modèles hybrides traditionnels combinant les cartes topologiques et les HMMs. Ce modèle hybride permet d'allier les points forts des cartes topologiques probabilistes et les modèles des HMMs pour la classification non supervisée de données séquentielles. Chaque cluster est représenté par un sous graphe de la carte permettant une initialisation du modèle HMM. Cette structure est ensuite optimisée par apprentissage du modèle HMM. L'apprentissage de chaque HMM permet un élagage des états redondants et par conséquent des cellules de la carte topologique. L'apport essentiel des deux approches que nous avons proposées est l'élaboration d'une topologie des modèles HMM. Dans la première

approche, on construit un HMM topologique sans aucun prétraitement des données. Par contre dans la deuxième approche la segmentation de l'espace des données par les cartes topologiques probabilistes permet de définir une notion de voisinage entre les différents modèles HMM.

Chapitre 1

Classification automatique

1.1 Introduction

La classification automatique peut être définie comme la division des données en groupes ou sous-ensembles d'objets similaires. Chaque groupe, appelé souvent "cluster", est composé d'objets similaires et sont dissimilaires avec les objets d'autres groupes. Représenter les données par un nombre de clusters implique nécessairement la perte de certains détails fins (c'est semblable à la compression de données avec perte d'information), mais on gagne en simplicité. Du point de vue de l'apprentissage automatique les clusters correspondent aux "données" cachées, le processus consistant à chercher des clusters est appelé apprentissage non-supervisé. La connaissance qui en résulte représente un concept de données. Par conséquent, le partitionnement "clustering" est l'apprentissage non-supervisé de concepts cachés des données. Les étapes principales de l'analyse et l'interprétation de la classification automatique sont représentés dans la figure 1.1. La fouille de données traite des grandes masses de données ce qui impose aux algorithmes de classification des contraintes par rapport au temps de calcul (complexité). Ces difficultés ont conduit à l'émergence d'assez puissantes méthodes de classification largement applicables dans le domaine de la fouille de données présentées par la suite.

1.1.1 Notations

Afin de fixer le contexte et la terminologie que nous allons utiliser, nous introduisons quelques notations. On considère un ensemble de données $\mathcal{A}$ composé des ob-

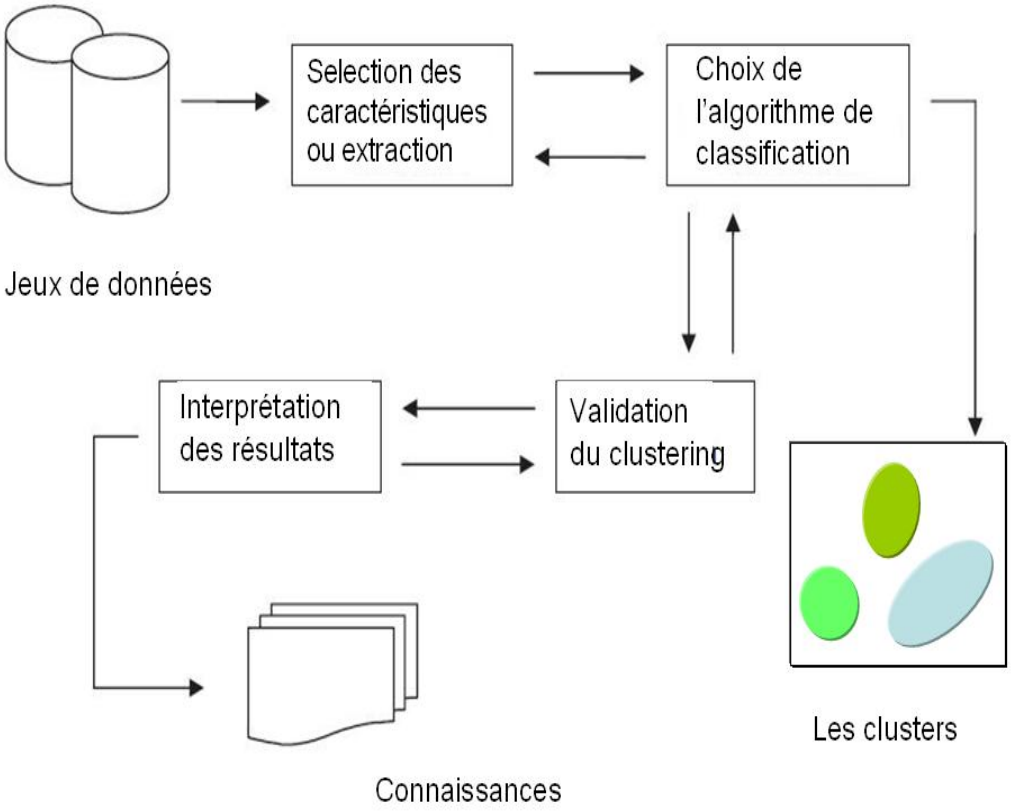


FIG. 1.1 – La procédure de classification. Le processus de base de l'analyse des clusters est composé de quatre étapes qui sont étroitement liées les unes aux autres et qui déterminent les clusters qui en résultent.

servations (ou autrement appellées, objets, instances, individus, transactions, etc.) $\mathbf{x}_i = (x_{i1}, ..., x_{id})$, où $i = 1 : N$, et chaque composante $x_{il} \in A_l$ est une variable continue ou catégorielle nominale (ou autrement appellées, *attribut, dimension, trait, champ, composante*). La combinaison d'observation variable forme un tableau de données qui conceptuellement correspond à une matrice $N \times d$ et est utilisé dans la majorité des algorithmes presentés ci-dessous. Cependant, des données d'un autre format, telles que des séquences de longueurs variables et des données hétérogènes, deviennent de plus en plus populaires. Ainsi l'objectif ultime du clustering est d'attribuer les observations à un système fini de k groupes (C_j). D'habitude, les sous-ensembles ne se chevauchent pas (cette contrainte est parfois violée), et leur union est égale à l'ensemble total des données avec des exceptions possibles de données atypiques ("outliers").

$$\mathcal{A} = C_1 \cup ... \cup C_r \cup ... \cup C_k \cup C_{atypiques}, \text{ où } C_r \cap C_k = \emptyset \tag{1.1}$$

1.1.2 Un survol bibliograhique

Les références générales sur le clustering sont variées et incluent [90, 78, 77, 96, 57, 119]. Une bonne introduction sur des techniques de classification utilisées dans le domaine du data mining peut être trouvée dans le livre [79]. Il existe un grand lien et une collaboration étroite entre les techniques d'apprentissage et d'autres disciplines (statistiques [7], bioinformatique [114]). Le domaine de la reconnaissance des formes est l'un des premiers domaines d'applications [51, 45, 68]. Des applications typiques incluent la reconnaissance de la parole et des symboles. Les algorithmes d'apprentissage numérique on été appliqués aussi à la *segmentation d'images*, *vision par machine* [89] et la réduction de dimensions dans le domaine de traitement d'images, qui est aussi connu comme *quantification vectorielle*. La classification a été aussi vue comme un problème d'estimation de densité, elle est équivalente dans ce cas à l'estimation statistique multivariée traditionnelle[141].

La classification automatique a trouvé différentes utilisations dans des domaines variés comme : la recherche et l'extraction d'information et la fouille de texte [42, 148, 47], dans des applications sur des données spatiales, par exemple, SIG (Système d'Information Géographique) ou sur des données astronomiques [156, 137, 55], l'analyse des données hétérogènes et séquentielles[34], les applications Web [147, 82], l'analyse ADN en bioinformatique [16], etc.

1.1.3 Caractéristiques des méthodes de classification

Les propriétés essentielles d'un bon algorithme de partitionnement sont :
- Indifférent à l'ordre des données en entrée
- Interpretabilité des résultats
- Capacité à gérer différents types de variables (attributs)
- Découverte de clusters avec des formes variables
- Incorporation de contraintes par l'utilisateur
- Passage à l'échelle
- Abilité de traiter des grandes bases de données
- Compléxité au niveau du temps
- Besoin minimum de connaissances du domaine pour déterminer les paramètres
- Prise en compte des "outliers“

Taxonomie des algorithmes de clustering

Une multitude d'algorithmes de classification sont proposés dans la littérature qui peuvent être groupés selon différents critères :

- Le type de données utilisées pour l'approche
- Le critère de classification qui définit la similarité entre les observations
- La théorie et les concepts de base sur lesquels les techniques de classification sont basées (par exemple la théorie fuzzy, les statistiques, les réseaux de neurones, ...)

Ainsi par rapport à la méthode utilisée pour définir les classes, les algorithmes peuvent être classifiés de façon générale dans les catégories suivantes [88] :

La classification hiérarchique. Le résultat de ce type d'algorithmes est un arbre de clusters, appelé dendogramme, qui montre comment les clusters sont organisés. En coupant le dendogramme au niveau désiré, une classification des données dans des groupes disjoints est ainsi obtenue.

La classification par partitionnement cherche à décomposer directement la base de données dans un ensemble de clusters disjoints. Plus spécifiquement, ils essaient de déterminer un nombre de partitions qui optimisent certains critères (fonction objective). La fonction objective peut mettre en valeur la structure locale ou globale des données et son optimisation est une procédure itérative.

La classification à base de densité. L'idée centrale de ce type de classification est de grouper des objets voisins de l'ensemble des données en classes suivant une fonction de densité.

La classification à base de grille. Ce type d'algorithmes est principalement proposé pour des données spatiales. La caractéristique principale de ce type d'algorithme est qu'il quantifie l'espace des données dans un nombre fini de cellules et ensuite il réalise toutes les opérations dans cet espace quantifié.

Autres types. Dans cette catégorie on attribue les algorithmes à base des modèles auto-organisés.

1.2 La classification hiérarchique

La classification hiérarchique construit une hiérarchie de clusters, ou autrement dit, un arbre de clusters, connue aussi sous le nom de dendogramme. Une telle approche permet d'explorer les données à travers différents niveaux de granularité. Les méthodes de la classification hiérarchique sont divisées en deux types d'approches : ascendante, dite *agglomérative* et descendante, dite *divisive* [96, 90] qui sont présentées dans la

figure 1.2.

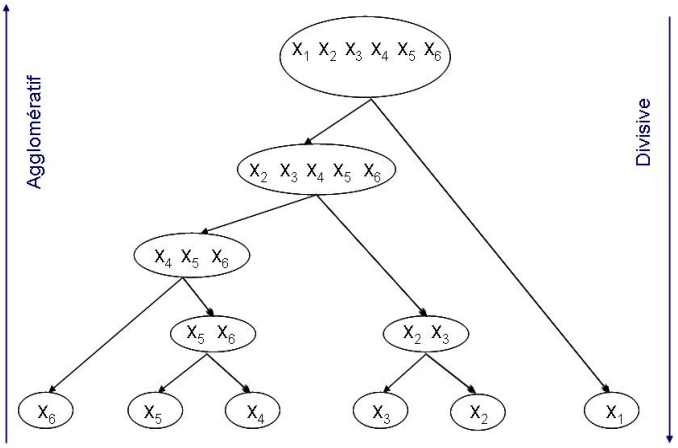


FIG. 1.2 – Classification hiérarchique.

La première (ascendante) qui est la plus couramment utilisée consiste, à construire l'hiérarchie à partir des objets (au départ on a un objet par classe), puis à fusionner les classes les plus proches, afin de n'en obtenir plus qu'une seule contenant tous les objets. La seconde (descendante), moins utilisée, consiste à construire l'hiérarchie à partir d'une seule classe regroupant tous les objets, puis à partager celle-ci en deux groupes. Cette opération est répétée à chaque itération jusqu'à ce que toutes les classes soient réduites à des singletons. Le processus continu jusqu'à ce qu'un critère d'arrêt (d'habitude c'est le nombre k de clusters demandé) est atteint. Les avantages de la classification hiérarchique incluent :

- Facilité pour traiter différentes formes de similarité ou de distance entre objets
- Applicable aux différents types d'attributs
- Une flexibilité en ce qui concerne le niveau de granularité

Les points faibles de la classification hiérarchique sont liés au :

- Choix du critère d'arrêt qui reste vague
- Fait que la plupart des algorithmes hiérarchiques ne revoient pas les clusters intermédiaires qu'une fois qu'ils sont construits pour les améliorer

Les méthodes de classification hiérarchique basées sur des métriques de liens donnent comme résultats des clusters de formes convexes (section 1.2.1). La section 1.2.3 contient des informations sur l'apprentissage incrémental, le clustering à base de modèles et le raffinement des clusters. Des algorithmes adaptés aux nuages de points sont présentés dans la section 1.2.2.

Dans la classification hiérarchique la représentation des données utilisée d'habitude, *observation-variable*, n'est pas très significative. Au lieu de cela on utilise une matrice de distance $N \times N$, qui calcule les similarités ou dissimilarités entre les données. Elle est parfois appelée matrice de *connectivité*. Les critères du lien décrits plus tard sont calculés à partir de cette matrice. Parfois, par rapport à la mémoire machine, la taille de la matrice des distances est trop grande. Pour relaxer cette limite, différentes heuristiques sont utilisées pour introduire dans cette matrice une densité plus faible. Cela peut être réalisé en omettant des entrées inférieures à un certain seuil, ou en utilisant seulement un certain sous-ensemble de représentants des données, ou en gardant pour chaque individu qu'un certain nombre des plus proches voisins. Une matrice creuse peut-être utilisée pour représenter intuitivement les concepts de connectivité et de proximité. On remarque que la manière dont on traite la matrice de (dis)simmilarité et on construit une métrique de liens reflète tout simplement nos idées a priori sur le modèle des donnèes.

A la matrice creuse on associe un graphe de connectivité $G = (X, E)$ où les noeuds X représentent les données et les arêtes E et leurs poids sont des paires de points et les entrées positives correspondant de la matrice. Ainsi, nous établissons le lien entre la classification hiérarchique et le partitionnement de graphe. La classification hiérarchique des données de grande taille peut être très sous-optimale, même si les données sont accessibles au niveau de la taille de la mémoire de la machine. Dans ce cas, la quantification des données peut améliorer la performance des algorithmes hiérarchiques.

Une des méthodes les plus populaires en classification hiérarchique est l'algorithme BIRCH (Balanced Iterative Reduction and Clustering using Hierarchies). Cet algorithme sauvegarde les informations de la classification dans un arbre balancé, où chaque entrée de l'arbre décrit un cluster et les nouveaux noeuds sont insérés sous l'entrée la plus proche. Un concept principal pour l'algorithme BIRCH est le CF-tree (Cluster Feature). Ainsi BIRCH dépend des paramètres qui contrôlent la construction de l'arbre CF tels que : le facteur de branchement, le maximum de points par feuille, le rayon des clusters et il dépend aussi de l'ordonnancement des données. Cet algorithme a une complexité de $O(N)$ et il contient plusieurs phases :

-La première

- Parcours de la base pour constriire un CF-tree initial en mémoire
- Une feuille est éclatée quand le seuil du diamètre est dépassé
- Les moments sont alors remontés aux noeuds internes

Cette phase permet d'obtenir un résumé comprimé de la base respectant les caracté-

ristiques de regroupement. Si l'arbre ne tient pas en mémoire on augmente le seuil du diamètre.

-La deuxième (optionnelle)

- Appliquer un algorithme de classification en mémoire aux noeuds feuilles du CF-tree, chaque noeud étant traité comme un point.
- Placer les points dans le cluster des centres les plus proches

1.2.1 Métrique des liens

La classification hiérarchique initialise un système de classes comme un ensemble de singletons (dans le cas agglomératif) ou comme un seul cluster qui contient toutes les données (dans le cas divisif) et procède itérativement par fusion ou éclatement des plus adéquats cluster(s) jusqu'à ce qu'un critère d'arrêt soit satisfait. La concordance d'un cluster pour la fusion/éclatement dépend de la (dis)similarité des éléments du cluster. Cela reflète l'hypothèse générale que les classes contiennent des points similaires. Pour fusionner/éclater des ensembles de points plutôt que des points à part, on doit généraliser la distance entre des points individuels à la distance entre sous-ensembles. De telles mesures de proximité dérivées, sont appelées **critères d'agrégation** ou **critères du lien (linkage metrics)**. Le type de métrique qu'on utilise affecte significativement les algorithmes hiérarchiques, parce qu'il reflète le concept de *connectivité* et de *proximité*. Les plus importantes métriques de liens inter-cluster [123, 129] sont : le critère du lien minimum (*single link*), critère du lien maximum *(complete link)* et critère du lien moyen *(average link)*. La mesure de dissimilarité sous-jacente est calculée pour chaque paire de points avec un point qui appartient au premier ensemble et un point dans le deuxième ensemble. Une opération spécifique telle que le minimum (*single link*), la moyenne *(average link)* et le maximum *(complete link)* :

$$d(C_1, C_2) = operation\{d(\mathbf{x}, \mathbf{y}) | \mathbf{x} \in C_1, \mathbf{y} \in C_2\}$$

Parmi les premières approches on retrouve l'algorithme SLINK [143], qui implémente le critère du lien minimum, la méthode du Voorhees [151], qui implémente le critère du lien moyen, et l'algorithme CLINK [44], qui utilise le critère du lien maximum. Parmi ceux-ci SLINK est le plus référencé. Il est lié au problème de trouver l'arbre couvrant basé sur la distance euclidienne minimale et il a une complexité quadratique en $O(N^2)$ [157]. Les méthodes qui utilisent les distances inter-cluster, et qui sont définies en fonction de paire de points appartenant à deux clusters (sous-ensembles) différents, sont appelées des méthodes de *graphe*. Ces méthodes de *graphe*, font allusion au graphe de connectivité $G = (X, E)$ présenté antérieurement, puisque chaque partition de données correspond à une partition de graphe. Telles méthodes peuvent être annexées par les

méthodes ***géométriques*** pour lesquelles chaque cluster est représenté par son point central. Pour les variables numériques, le point central est défini comme un centroïde ou une "moyenne" de deux centroïdes des clusters susceptibles de former une agglomération. Toutes les métriques de liens présentées précédemment peuvent être dérivées par exemple de la mise à jour de la formule de Lance-Williams [101].

$$d(C_i \cup C_j, C_k) = a(i)d(C_i, C_k) + a(k)d(C_j, C_k) + bd(C_i, C_j) + c|d(C_i, C_k) - d(C_j, C_k)|$$

où a, b, c sont des coefficients qui correspondent à un type de lien.

Cette formule exprime un lien métrique entre l'union de deux clusters avec un troisième cluster en terme de composantes sous-jacentes. La formule de Lance-Williams a une grande importance puisqu'elle rend les calculs par rapport à la (dis)similarité faisable. Une étude sur les critères du lien peut être trouvée dans [124, 43]. La classification hiérarchique souffre par rapport à la complexité du temps de calcul. D'habitude les méthodes basées sur les métriques de liens ont une complexité en $O(N^2)$ [129]. Malgré la complexité défavorable, ces algorithmes sont largement utilisés.

1.2.2 Classification hiérarchique de formes variables

Les métriques basées sur la distance euclidienne pour la classification hiérarchique des données spatiales sont naturellement prédisposées pour les clusters aux formes convexes. Mais, par exemple, l'examination visuelle des images spatiales atteste fréquemment des clusters aux apparences curvilignes. Guha et al. [75] ont introduit l'algorithme hiérarchique agglomeratif CURE (Clustering Using REpresentatives). Cet algorithme apporte de nouvelles caractéristiques d'un intérêt général. Il a une attitude à part pour les données atypiques ("outliers") et pour l'étape d'attribution des étiquettes. Il utilise aussi deux manières pour effectuer le passage à l'échelle. La première c'est l'échantillonnage des données. La deuxième manière c'est le partitionnement des données en K sous-ensembles, telle que les clusters d'une granularité fine sont construits d'abord dans des partitions. Un point important de CURE est qu'il représente un cluster par un nombre c fixe de points dispersés autour de lui. La distance (entre deux clusters) utilisée dans le processus aggmomeratif est égale au minimum de la distance entre deux représentants dispersés. Par conséquent, CURE adopte une voie du milieu, entre les approches basées sur les graphes (tous les points) et les méthodes géométriques (un centroïde). Les liens de proximité maximale ou moyenne sont remplacés par des représentants de proximité globale. En choisissant des représentants autour d'un cluster, on arrive a couvrir des formes non sphériques. Comme dans les cas précédents, le processus d'agglomération continue tant que le nombre de clusters K n'est pas atteint.

L'algorithme CURE utilise une astuce supplémentaire : les points dispersés qui sont sélectionnés sont concentrés autour du centroïde géométrique du cluster à l'aide d'un paramètre α spécifié par l'utilisateur. Le processus de concentration enlève l'influence des "outliers" dans le cas où les "outliers" sont localisés plus loin par rapport au centroïde du cluster que les autres représentants dispersés. CURE est capable de trouver des clusters de différentes formes et dimensions et il est insensible aux "outliers". Comme CURE utilise l'échantillonnage, l'estimation de sa complexité n'est pas évidente. Pour des données de petites dimensions les auteurs fournissent une complexité estimée à $O(N^2)$ définit en terme de dimension de l'échantillon. Plus exactement, les bornes dépendent des paramètres d'entrée : le paramètre de concentrantion α, le nombre c de points représentatifs, le nombre des partitions et la taille de l'échantillon.

Tandis que l'algorithme CURE opère avec des attributs numériques (en particulier des données spatiales de petites dimensions), l'algorithme ROCK développé par les mêmes chercheurs [76] vise le clustering hiérarchique aglommeratif pour de données catégorielles. L'algorithme hiérarchique agglomératif CHAMELEON [93] utilise la modélisation dynamique dans l'agrégation des clusters. Il utilise le graphe de connectivité G qui correspond à la matrice creuse de connectivité obtenue par l'approche des k plus proches voisins, d'une telle manière que les arêtes des k plus similaires points vers d'autres points sont préservés, et le reste est élagué. CHAMELEON est composé de deux étapes : dans la première étape des petits clusters sont générés pour l'initialisation de la deuxième étape. Cela implique un partitionnement de graphe [94]. Dans la deuxième étape, le processus d'agglomération est réitéré. Il utilise des mesures d'inter-connectivité relative $RI(C_i, C_j)$ et proximité relatif $RC(C_i, C_j)$; les deux sont normalisées localement par les quantités liées aux clusters C_i, C_j. De ce point de vue la modélisation est *dynamique*. Le processus d'agglomération dépend des seuils fournis par l'utilisateur. La décision de fusionner est réalisée en se basant sur la combinaison des mesures relatives locales comme suit :

$$I(C_i, C_j) = RI(C_i, C_j) \cdot RC(C_i, C_j)^{\alpha}$$

Cet algorithme fournit des clusters de différentes dimensions, formes, et densités dans un espace bidimensionnel. Il a une complexité en $O(Nm + Nlog(N) + m^2 \log(m))$ où m est le nombre de sous-classes construites pendant la phase d'initialisation.

1.2.3 Autres critères d'agglomération

La méthode de Ward [155] implémente un clustering agglomératif basé non pas sur une métrique de liens, mais sur une fonction objective utilisée en k-means (sous-section 1.3.3). La décision de fusion est prise en fonction de ses impacts sur la fonction objective. L'algorithme hiérarchique populaire COBWEB [60], pour les données catégorielles, a deux qualités importantes. La première c'est qu'il utilise l'apprentissage *incrémental*. Au lieu d'utiliser des approches divisives ou agglomératives, il construit dynamiquement un dendogramme en traitant une observation à un moment donné. La deuxième, COBWEB appartient à l'apprentissage *conceptuel* ou *basé sur modèles*. Ça veut dire que chaque cluster est considéré comme un modèle qui peut être décrit intrinsèquement, plutôt qu'une collection de points qui lui sont affectés. Le dendogramme du COBWEB est appellé un *arbre de classification*. Chaque noeud C de l'arbre (un cluster), est associé à la probabilité conditionnelle pour les paires catégorielles attribut-valeurs :

$$Pr(x_l = v_{lp}|C), l = 1 : d, p = 1 : |A_l|. \tag{1.2}$$

Ça peut être reconnu facilement comme un classifieur Naïve Bayes C-spécifique. Durant la construction de l'arbre de classification, chaque nouveau point est descendu le long de l'arbre et l'arbre est potentiellement mis à jour (par une opération d'insertion/fusion/éclatement/création). La décision est basée sur l'analyse de *l'utilité de catégorie* (UC) définie de la manière suivante :

$$UC\{C_1, ..., C_k\} = \frac{\left(\sum_{j=1:k} UC(C_j)\right)}{k},$$

$$UC(C_j) = \sum_{l,p}(Pr(x_l = v_{lp}|C_j)^2 - (Pr(x_l = v_{lp})^2))$$

Similaire à l'index GINI, elle permet aux clusters C_j d'améliorer la prédiction des valeurs des attributs catégoriels v_{lp}. Etant incrémental, COBWEB a une complexité en $O(tN)$, quoiqu'il dépend d'une manière non-linéaire des caractéristiques de l'arbre qui sont gardées dans la constante t. Il existe aussi un algorithme incrémental similaire pour des attributs numériques appelé CLASSIT [70]. CLASSIT associe des distributions normales avec les noeuds du cluster. Les deux algorithmes peuvent entrainer des arbres fortement déséquilibrés.

Chiu et al. [38] ont proposé une autre approche conceptuelle pour la classification hiérarchique. Ce développement contient plusieurs caractéristiques utiles, telle que l'extension de la phase de prétraitement de l'algorithme BIRCH aux attributs catégoriels,

la manipulation des "outliers", et une stratégie à deux étapes pour contrôler le nombre de clusters y compris BIC (défini dans la section 1.7.2). Le modèle associé à un cluster couvre les deux types d'attributs : numérique et catégorielle et constitue un mélange de modèles gaussiens et multinomiaux. On note les paramètres multivariables qui leurs correspond par θ. A chaque cluster C, on associe le logarithme de sa vraisemblance :

$$V_C = \sum_{\mathbf{x}_i \in C} \log p(\mathbf{x}_i|\theta)$$

L'algorithme utilise *l'estimation du maximum de la vraisemblance* pour le paramètre θ. La distance entre deux clusters est définie (à la place de la métrique des liens) comme une décroissance de la log-vraisemblance causée par la fusion des deux clusters considérés.

$$d(C_1, C_2) = V_{C_1} + V_{C_2} - V_{C_1 \cup C_2} \tag{1.3}$$

Le processus d'agglomération continue jusqu'à ce qu'un critère d'arrêt soit satisfait. Ainsi, la détermination des meilleurs p est automatique. La complexité de l'algorithme est linéaire en N pour la phase de synthèse. La classification hiérarchique traditionnelle est inflexible dû à son approche gloutonne : après qu'une fusion ou une division soit sélectionnée, il n'est pas raffiné. Fisher [61] a étudié la redistribution des clusters par la méthode hiérarchique itérative pour améliorer le dendogramme une fois construit. Karypis et al. [69] ont cherché aussi un raffinement pour la classification hiérarchique. Pour des références liées à l'implémentation parallèle de la classification hiérarchique, le lecteur pourra consulter [129].

Dans le tableau 1.1 nous avons résumé et comparé les caractéristiques générales des plus représentatifs algorithmes décrits dans cette section. Plus spécifiquement la comparaison est basée sur les caractéristiques (propriétés) suivantes des algorithmes : 1) le type de données que l'algorithme supporte (numérique, catégoriel où spatial), 2) la forme des clusters, 3) la capacité de traiter le bruit et les données atypiques ("outliers"), 4) les paramètres d'entrée, 5)les résultats de la classification.

Méthode	***Type des données***	***Complexité***	***Géométrie***	***"Outliers"***	***Paramètres d'entrée***	***Résultats***
BIRCH	Numérique	$O(N)$	Formes non-convexes	Oui	Le rayon des clusters, le facteur de branchement	CF=(le nombre de points dans le cluster, la somme linéaire des points dans le cluster LS, la somme carrée des N points SS)
CURE	Numérique	$O(N^2 \log N)$	Formes arbitraires	Oui	Le nombre des clusters, le nombre des représentants des clusters	L'affectation des valeurs de données aux clusters
ROCK	Catégorielle	$O(N^2 + Nm_m m_a + N^2 \log N)$, $O(N^2 \log N)$ où m_m est le nombre maximal des voisins pour un point et m_a est le nombre moyen de voisins pour un point	Formes arbitraires	Oui	Le nombre des clusters	L'affectation des données aux clusters

TAB. 1.1 – Les caractéristiques principales des algorithmes de classification hiérarchique (où N est le nombre des observations considérées).

1.3 La classification non hiérarchique

Dans cette section nous allons présenter les algorithmes de partitionnement de données, qui partagent les données en plusieurs sous-ensembles. Puisque l'examen de tous les sous-ensembles possibles est infaisable du point de vue computationnel, quelques heuristiques gloutonnes sont utilisées sous forme d'*optimisation itérative*. Plus précisément, cela correspond aux différents schémas de *réallocation* qui réaffectent itérativement les points entre les k clusters. Par rapport aux méthodes hiérarchiques traditionnelles, dans lesquelles les clusters ne sont pas revus après avoir été construits, les méthodes par réaffectations améliorent les clusters progressivement.

Une des voies pour réaliser le partitionnement de données, c'est d'avoir un point de vue *conceptuel* qui identifie le cluster avec un certain modèle dont les paramètres inconnus doivent être estimés. Plus exactement, les modèles *probabilistes* supposent que les données proviennent d'un mélange de plusieurs populations dont on ne connait pas les distributions et d'autres paramètres à estimer. Ces algorithmes sont décrits dans la section 1.3.1. Un grand avantage des méthodes probabilistes est l'interpretabilité des classes obtenues. Une autre voie commence avec la définition d'*une fonction objective* (coût,énergie...) qui dépend de la partition. Comme on l'avait vu dans la sous-section 1.2.1, la distance par paire ou la similarité peut être utilisée pour calculer des relations inter et intra-cluster. Dans le cas de l'amélioration itérative, des calculs par paire sont trop coûteux. L'utilisation d'un représentant unique du cluster résoud le problème ; le calcul de la fonction objective devient linéaire en N(le nombre de clusters $k \ll N$).

En fonction de la méthode de calcul des prototypes, les méthodes de partitionnement par optimisation itérative sont divisées en *k-médoïdes* et *k-moyennes*. K-médoïde a deux avantages : le premier, est qu'il n'a pas de limites par rapport au type d'attributs et deuxièmement, le choix du medoïde est dicté par la location d'une fraction des points prédominants dans le cluster et ainsi est moins sensible à la présence des données aberrantes. Dans le cas des $k - moyennes$, un cluster est représenté par son centre, qui est "une moyenne" (d'habitude c'est une moyenne ponderée) des points affectés au cluster. Ce type d'algorithme est très adapté aux cas d'attributs numériques, mais reste sensible à la présence du bruit et d'outliers. D'une autre part, les centroïdes ont l'avantage d'avoir un sens géométrique et statistique assez clair. Ces approches sont décrites dans les sections 1.3.2, 1.3.3.

1.3.1 Classification probabiliste

Dans les approches *probabilistes*, les données sont considérées comme un échantillon extrait indépendamment d'un *modèle de mélanges* de plusieurs distributions de probabilités [116]. La région autour de la moyenne de chaque distribution (supposé unimodale) constitue un cluster naturel. Donc le cluster est associé aux paramètres de la distribution telle que : la moyenne, l'écart-type, etc. Chaque observation contient non seulement ses attributs (observables), mais aussi l'étiquette du cluster. Différentes méthodes sont utilisées pour faciliter la découverte d'un meilleur optimum local. Moore [121] a suggéré l'accélération de la méthode EM basé sur un indice spécial de données, KD-tree. Les données sont divisées à chaque noeud en deux descendant par l'éclatement du plus grand attribut au centre de son écart. Chaque noeud garde suffisamment de statistiques (y compris la matrice de covariance) similaires à BIRCH. Un calcul approximatif sur un arbre trié accélère les itérations de EM. La classification probabiliste a plusieurs caractéristiques importantes :

- Elle peut être modifiée pour recoder les structures complexes
- A n'importe quelle étape du processus itératif le modèle de mélange intermédiaire peut être utilisé pour affecter des cas (des propriétés émergentes en ligne)
- Elle a comme résultats un système de classes facilement interprétables

Puisque les modèles de mélanges ont une base probabiliste solide, la détermination du nombre le plus approprié des classes devient une tâche plus réalisable. Du point de vue de la fouille de données, un nombre excessif de paramètres peut causer du surapprentissage, tandis que d'un point de vue probabiliste, le nombre des paramètres peut être traité dans le cadre Bayesian. L'algorithme AUTOCLASS [36] utilise aussi un modèle de mélange et couvre une grande variété de distributions, telles que la distribution de Bernoulli, Poisson, gaussienne et log-normale. L'algorithme MCLUST [63] est une approche pour la classification hiérarchique, à base de modèles de mélange et l'analyse discriminante utilisant le critère BIC (voir la section 1.7.2). MCLUST utilise des modèles Gaussiennes avec des ellipsoïdes de différents volumes, formes, et orientations.

Une caractéristique importante de la classification probabiliste est que les modèles de mélange peuvent être généralisées naturellement pour la classification de données "hétérogènes". La classification à base de modèles est aussi utilisée dans le cadre hiérarchique : les algorithmes qui en font partie COBWEB, CLASSIT et les méthodes developpées par Chiu et al. [38] ont été présentées antérieurement.

1.3.2 Les méthodes K-medoïdes

Dans les méthodes k-médoïdes une classe est représentée par un de ses points. Nous avons déjà mentionné que c'est un algorithme efficace puisqu'il traite n'importe quel type d'attributs et que les medoïdes ont une résistance intégrée contre les données atypiques puisque les points périphériques des clusters ne sont pas pris en compte. Quand les medoïdes sont sélectionnés, les classes sont définies comme des sous-ensembles de points proches des medoïdes respectifs, et la fonction objective est définie comme la distance moyenne ou une autre mesure de dissimilarité entre un point et son medoïde. Parmi les premières versions des méthodes k-médoïdes nous pouvons citer l'algorithme PAM (Partitioning Around Medoids) et l'algorithme CLARA (Clustering LARge Applications)[96]. PAM est une optimisation itérative qui combine la réaffectation des points entre les clusters avec la re-nomination des points comme des médoïdes potentiels. Le principe pour ce processus est l'effet sur la fonction objective, qui, évidemment, est une stratégie assez coûteuse. CLARA utilise plusieurs échantillons, chacun avec $40+2k$ points, qui sont soumis à PAM. Tout l'ensemble des données est affecté aux médoïdes résultants, le meilleur système de médoïdes est retenu.

Ng et Han [126] ont introduit l'algorithme CLARANS (Clustering Large Applications based upon RANdomized Search) dans le contexte de la classification des données spatiales. Les auteurs proposent de construire un graphe dont les noeuds représentent les ensembles des k médoïdes et une arête lie deux noeuds s'ils diffèrent exactement avec un noeud. Tandis que CLARA compare très peu de voisins qui correspondent à un petit échantillon fixé, CLARANS utilise une recherche aléatoire pour générer les voisins en commençant avec un noeud arbitraire et en vérifiant aléatoirement le paramètre "maxneighbor" des voisins. Si un voisin représente une partition meilleure, le processus continu avec ce nouveau noeud. Sinon un minimum local est calculé, et l'algorithme recommence jusqu'à ce qu'un *numlocal* est trouvé (la valeur de *numlocal*=2 est recommandée). Le meilleur noeud (l'ensemble de médoïdes) est retourné pour former les partitions résultantes. La complexité de CLARANS est en $O(N^2)$ en terme de nombre d'observations. Ester et al. [53] ont adapté l'algorithme CLARANS aux données spatiales de très grandes dimensions. Ils ont utilisé le R^*-tree [15] pour relaxer la contrainte originale que toutes les données résident dans la mémoire centrale, ce qui permet de concentrer l'exploration sur la partie relevante de la base de données qui réside dans la branche de tout l'arbre de données.

1.3.3 Les méthodes K-moyennes

L'algorithme **k-moyennes** ou "k-means"[77, 81] est l'outil de classification le plus utilisé dans les applications scientifiques et industrielles. Quoiqu'il ne fonctionne pas très bien pour les attributs catégoriels, il a un bon sens géométrique et statistique pour les attributs numériques. La somme des dispersions (distances) entre un point et son centroïde, exprimée par une distance appropriée, est utilisée comme fonction objective. Par exemple, la fonction objective basée sur la norme L_2, la somme des carrées des erreurs entre les points $\mathbf{x}_i$ et les centroïdes $\mathbf{w}_j$ correspondant, est égale à la variance intra-cluster totale :

$$E(C) = \sum_{j=1:K} \sum_{\mathbf{x}_i \in A} \|\mathbf{x}_i - \mathbf{w}_j\|^2 \tag{1.4}$$

La somme des carrés des erreurs peut être rationalisée comme le log-vraisemblance pour les modèles de mélanges distribués normalement. Par conséquent, k-means peut être dérivé du formalisme probabiliste général (voir la sous-section 1.3.1) [120]. On rappelle que seulement les moyennes sont estimées (dans le cas des k-moyennes) . Une fonction objective basée sur la norme L_2 a plusieurs proprietés algébriques uniques. Elle coïncide avec l'erreur par paire et avec la différence entre la variance totale des données et la variance inter-classes :

$$E^{'}(C) = \frac{1}{N} \sum_{j=1:K} \sum_{\mathbf{x}_i, \mathbf{y}_i \in A} \|\mathbf{x}_i - \mathbf{y}_i\|^2, \tag{1.5}$$

Par conséquent, la séparation des classes est atteinte en même temps que la consistance du cluster. Il existe deux versions d'optimisation itérative du k-moyennes.

La première version est similaire à l'algorithme EM et consiste à réitérer les deux étapes suivantes : (1) réaffecter tous les points à leurs centroïdes proches, et (2) recalcule les centroïdes des nouveaux groupes formés. Les itérations continuent jusqu'à ce qu'un critère d'arrêt est atteint (par exemple, le nombre d'itérations). Cette version est connue comme l'algorithme de Forgy [62] et a plusieurs avantages :

- Il travaille facilement avec n'impotre quelle norme L_p
- Il permet une simple parallélisation [49]
- Il est insensible à l'ordre des données

La deuxième version du k-moyennes (classique dans l'optimisation itérative) réaffecte des points basés sur l'analyse plus détaillée des effets sur la fonction objective causés par le mouvement d'un point de son groupe actuel vers un nouveau cluster potentiel. Si le transfert a un effet positif, le point est déplacé et les deux centroïdes sont recalculés. Cette version est difficilement réalisable du point de vue computationnel, car elle

nécessite une boucle propre pour chaque observation par déplacement des centroïdes. Cependant, dans le cas L_2 [51, 19] tous les calculs peuvent-être algébriquement reduits à calculer simplement une seule distance. Par conséquent, dans ce cas les deux versions ont la même complexité de calcul. C'est une évidence expérimentale que la deuxième version (classique), comparée à algorithme de Forgy, fournit souvent de meilleurs résultats [102, 148]. En particulier, Dhillon et al. [48] ont remarqué que les *k-moyennes* sphériques de Forgy (utilisant une similarité basée sur le cosinus à la place d'une distance Euclidienne) ont tendance de bloquer quand ils sont appliqués à la classification de documents. Ils se sont aperçus qu'une version qui réaffecte les observations et recalcule immédiatement les centroïdes fonctionne beaucoup mieux. En plus de ces deux versions, il y a eu plusieurs tentatives pour trouver le minimum de la fonction de coût des *k-moyennes*. Dans l'algorithme ISODATA [10] les auteurs ont utilisé la fusion ou l'éclatement des clusters intermédiaires.

La grande popularité des *k-moyennes* est bien méritée. C'est un algorithme simple qui se base sur un fondement solide de l'analyse de la variance, mais il souffre des aspects suivants :

- Les résultats dépendent beaucoup de l'initialisation
- Le minimum local calculé semble être très loin du minimum global
- Le processus est sensible aux données atypiques ("outliers")
- Les classes qui résultent peuvent être déséquilibrées (même vide, dans la version de Forgy)

Un moyen simple d'atténuer l'effet de l'initialisation des clusters a été suggéré, par [33]. Dans ce cas l'algorithme *k-moyennes* est exécuté sur plusieurs petits échantillons de données avec une affectation initiale aléatoire. Chacun de ses systèmes construits est utilisé comme initialisation possible pour l'ensemble de tous les échantillons. Les centroïdes du meilleur système construit de cette manière sont conseillés pour lancer l'algorithme *k-moyennes* sur tout l'ensemble de données. Aucune initialisation actuelle ne garantit un minimum global pour *k-moyennes*.

Zhang [159] a suggéré une autre manière d'améliorer le processus d'optimisation par une affectation douce (soft) des points aux différents groupes avec des poids appropriés (comme EM le fait), plutôt que de les déplacer définitivement d'un cluster à un autre. Une méthode générique pour atteindre la mise à l'échelle est de prétraiter ou diviser les données. Un tel prétraitement d'habitude prend en compte les données atypiques. Le prétraitement a ses inconvénients. Il s'agit d'une approximation qui parfois affecte négativement la qualité du cluster final. Pelleg et Moore [130] ont suggéré une approche pour accélérer le processus itératif des *k-moyennes* (sans utiliser la division de la base),

en utilisant KD-tree [122]. L'algorithme X-means [131] va plus loin : en plus de l'accélération du processus itératif, il essaie d'intégrer une recherche pour les meilleurs k dans le processus lui-même. X-means cherche à diviser une partie des groupes déjà construits en se basant sur le critère BIC. Ainsi, une meilleure initialisation est obtenue pour les itérations suivantes qui couvrent un spectre de k groupes admissibles spécifiés par l'utilisateur.

La grande popularité des *k-moyennes* a favorisé l'apparition de plusieurs extensions et modifications. La distance de Mahalanobis peut être utilisée pour couvrir des clusters hyper-ellipsoïdals [110]. On peut utiliser comme fonction de coût le maximum de la variance intra-classes, au lieu de la somme [73]. Des adaptations aux données catégorielles sont connues [111]. Dans ce contexte on utilise parfois le terme k-prototypes [85]. Dans le tableau 1.3.3 nous avons résumé et comparé les caractéristiques générales des plus représentatifs algorithmes décrits dans cette section.

Méthode	***Type des données***	***Complexité***	***Géométrie***	***Outliers, bruit***	***Paramètres d'entrée***	***Résultats***
K-moyennes	Numérique	$O(N)$	Formes non-convexes	Non	Le nombre des clusters	Le centre des clusters
K-mode	Catégorielle	$O(N)$	Formes non-convexes	Non	Le nombre des clusters	Les modes des clusters
PAM	Numérique	$O(k(N-k)^2)$	Formes non-convexes	Non	Le nombre des clusters	Les médoïdes des clusters
CLARA	Numérique	$O(k(40+k)^2+k(N-k))$	Formes non-convexes	Non	Le nombre des clusters	Les médoïdes des clusters
CLARANS	Numérique	$O(kN^2)$	Formes non-convexes	Non	Le nombre des clusters, le nombre maximal des voisins examinés	Les médoïdes des clusters

TAB. 1.2 – Les caractéristiques principales des algorithmes de classification non hiérarchique (où N est le nombre des observations considérées).

1.4 Les méthodes à base de densité

Un ensemble ouvert dans l'espace Euclidien peut être divisé en un ensemble de ses composantes connectées. L'implémentation de cette idée pour le partitionnement d'un ensemble fini de points a besoin de tels concepts comme la densité, la connectivité et les limites (extrémité, bordure, frontière). Ils sont fortement reliés au plus proches voisins d'un point. Un cluster, défini comme une composante densément connectée, évolue dans n'importe quelle direction où la densité le mène. Par conséquent, les algorithmes à base de densité sont capables de découvrir des clusters de formes arbitraires. Ainsi, ce principe fournit une protection naturelle contre les données aberrantes. La figure 1.3 nous montre quelques formes de clusters qui présentent un problème pour la classification avec réaffectation (telle que k-moyennes), mais qui sont traités avec succès par les algorithmes à base de densité.

D'un point de vue général par rapport à la description des données, un seul cluster

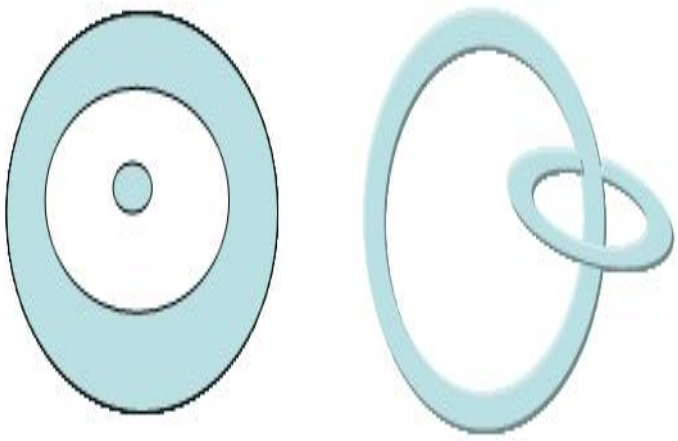


FIG. 1.3 – Des formes irrégulières difficiles à traiter pour k-moyennes

dense, composé de deux zones adjacentes avec des densités significativement différentes (les deux supérieures à un seuil) n'est pas très informatif. On trouve une bonne introduction sur les algorithmes à base de densité dans le livre de [79]. Puisque les algorithmes à base de densité ont besoin d'une métrique de l'espace, un bon cadre pour eux sera la classification des données spatiales [80, 100]. Les indices classiques sont effectifs seulement avec des données de petites dimensions. L'algorithme DENCLUE, qui, en effet, est un mélange entre la classification à base de densité et un prétraitement à base de grille, est moins affecté par la dimensionnalité des données. Il existe deux types d'approches à base de densité. Le premier type d'approche attache la densité a un point qui appartient aux données d'apprentissage et est décrit dans la sous-section 1.4.1. Il compte des algorithmes spécifiques comme DBSCAN, GDBSCAN, OPTICS et

DBCLASD. Le deuxième type d'approche attache la densité à un point dans l'espace d'attributs et est expliqué dans la sous-section 1.4.2. Il inclut l'algorithme DENCLUE.

1.4.1 La connectivité pour les méthodes à base de densité

Des concepts majeurs de cette section sont la **densité** et la **connectivité**, les deux mesurées en terme de distribution locale des plus proches voisins. L'algorithme DBSCAN (Density Based Spatial Clustering of Applications with Noise) [54] qui vise les données spatiales de petites dimensions est le plus connu dans cette catégorie. Deux paramètres d'entrée ε et *MinPts* sont utilisés par l'algorithme :

1) Un **ε-voisinage** $N_\varepsilon(\mathbf{x}) = \{y \in \mathcal{A} | d(\mathbf{x}, \mathbf{y}) \leq \varepsilon\}$ du point $\mathbf{x}$
2) Un **objet central** (un point avec un voisinage de plus de *MinPts* points)
3) Un concept d'un point $\mathbf{y}$ qui est **accessible au niveau de la densité** par un objet central $\mathbf{x}$ (il existe une séquence finie d'objets centraux entre $\mathbf{x}$ et $\mathbf{y}$ tel que chaque successeur appartient à un ε-voisinage de ses prédécesseurs)
4) Une connectivité de densité de deux points $\mathbf{x}$, $\mathbf{y}$ (ils doivent être accessible par la densité par un objet central en commun).

Ainsi la connectivité de la densité est une relation symétrique, et tous les points joignables par les objets centraux peuvent être factorisés dans des composantes connectées au maximum qui peuvent être interpretées comme des clusters. Les points qui ne sont pas connectés à aucun point central sont des "outliers" (ils ne sont pas couverts par aucun cluster). Les points qui ne sont pas centraux dans un cluster représentent ses *frontières*. Finalement, les points centraux sont des points *internes*. Le prétraitement est indépendant du classement(rangement) des données. Jusqu'ici, rien ne demande des limitations sur la dimension ou le type d'attribut. Certes, un calcul efficace du ε-voisinage présente un problème. Néanmoins, dans le cas des données **spatiales** de petites dimensions, différents schémas d'indexation effective existent. L'algorithme DBSCAN repose sur l'arbre d'indexation R^*-tree [15]. Par conséquent, la complexité théorique du DBSCAN sur des données spatiales de petites dimensions est de $O(N \log(N))$. Les expériences attestent des temps d'exécution super-linéaire. On remarque que DBSCAN repose sur le ε-voisinage et la fréquence du comptage. Beaucoup de données spatiales contiennent au lieu des points, des objets étendus tels que les polygones. Des mesures supplémentaires (tel que l'intensité d'un point) peuvent être utilisées à la place d'un simple comptage. Ces deux généralisations mènent à l'algorithme GDBSCAN [137], qui utilise les deux parcmètres comme DBSCAN. Par rapport à ces deux paramètres ε et *MinPts*, il n'y a pas de moyen simple pour les adapter aux données. En outre, différentes parties des données peuvent exiger différents paramètres - le problème discuté

précédemment en liaison avec CHAMELEON(section 1.2).

L'algorithme OPTICS (Ordering Points To Identify the Clustering Structure)[[4] adapte DBSCAN pour ces défits. Il construit une classification augmentée des données qui est compatible avec DBSCAN, mais en gardant les deux paramètres ε, *MinPts*, OPTICS couvre un spectre de tous les $\varepsilon \leqslant \varepsilon^{'}$ différents. La classification construite peut être utilisée automatiquement ou interactivement. Avec chaque point, OPTICS garde uniquement deux champs supplémentaires, la distance centrale ("core-distance") et la distance d'accessibilité ("reachability-distance"). Par exemple, la distance centrale est la distance vers le plus proche voisin de *MinPts* quand il ne dépasse pas ε, ou indéfini sinon. Expérimentalement, OPTICS montre un temps d'exécution approximativement égal au temps d'exécution du DBSCAN.

Tandis que OPTICS peut être considéré comme une extension du DBSCAN dans le sens de différentes densités locales, une approche plus solide du point de vue mathématique est d'estimer une variable aléatoire égale à la distance entre un point et son plus proche voisin, et d'apprendre sa probabilité de distribution. Au lieu de se baser sur les paramètres définis par l'utilisateur, une conjoncture possible est que chaque cluster ait sa propre échelle typique qui détermine la distance vers le plus proche voisin. Le but est de découvrir de telles échelles. Une telle approche non-paramétrique est implémentée dans l'algorithme DBCLASD (Distribution Based Clustering of Large Spatial Database)[156]. Supposant que les points dans chaque groupe sont uniformement distribués, ce qui peut être réaliste ou pas, DBCLASD défini un cluster comme un sous-ensemble de X d'une forme arbitraire non-vide, qui a la distribution attendue de la distance vers le plus proche voisin avec une confiance requise, et c'est l'ensemble *connecté au maximum* avec cette qualité. Cet algorithme traite les données spatiales. Le test χ^2 est utilisé pour vérifier les conditions de la distribution (une conséquence standard est l'exigence pour chaque cluster d'avoir au moins 30 points). Par rapport à la connectivité, DBCLASD se repose sur les approches *à base de grille* pour générer les polygones qui approximent les clusters. L'algorithme possède les moyens pour traiter des bases de données réelles avec du bruit et implémente l'apprentissage non supervisé *incremental*. L'affectation des points n'est pas définitive. Certains points (du bruit) ne sont pas affectés, mais ils sont analysés plus tard. Par conséquent, les points explorés incrementalement peuvent être vérifiés d'une manière intrinsèque. DBCLASD est connu être 60 fois plus rapide que CLARANS sur certains jeux de données. Par rapport à DBSCAN, il peut être 2-3 fois plus lents. Cependant, DBCLASD n'exige pas des paramètres de l'utilisateur, alors que la recherche empirique des paramètres adéquats demande plusieurs exécutions de DBSCAN. En plus, DBCLASD découvre

des clusters de densités différentes.

1.4.2 La fonction de densité

Hinneburg et Keim [84] ont calculé des fonctions de densité définies sur l'espace des attributs sous-jacent. Ils ont proposé l'algorithme DENCLUE (DENsity-based CLUstEring). Comme DBCLASD, il a des bases mathématiques solides. DENCLUE utilise une fonction de densité définie comme suite :

$$f^{\mathcal{A}}(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{A}} f(\mathbf{x}, \mathbf{y}) \tag{1.6}$$

qui est la superposition de plusieurs fonctions d'influence. Quand les termes f dépendent de $\mathbf{x} - \mathbf{y}$, la formule peut être reconnue comme une convolution avec un noyau. Des exemples incluent une *fonction carré* $f(\mathbf{x}, \mathbf{y}) = \theta(\|\mathbf{x} - \mathbf{y}\|)/\sigma$ égale à 1, si la distance entre $\mathbf{x}$ et $\mathbf{y}$ est plus petite ou égale à σ, et une fonction Gaussienne d'influence $f(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$. Cela fournit un haut niveau de généralisation : le premier exemple mène à DBSCAN (section 1.4.1), et le deuxième aux k-means (section 1.3.3). Les deux exemples dépendent du paramètre σ. En limitant la sommation à $A = \{\mathbf{y} : \|\mathbf{x} - \mathbf{y}\| < k\sigma\} \subset X$ nous permet une implementation pratique.

L'algorithme DENCLUE est concentré sur des maximums locaux de fonctions de densité appelées attracteurs de densité et utilise une série de techniques descendantes et ascendantes de gradients pour les trouver. En plus de ces clusters définis par rapport au centre, les clusters de formes arbitraires sont définis comme une continuation le long des séquences de points dont les densités locales ne sont pas plus petites que le seuil prévu ξ. L'algorithme est stable par rapport aux "outliers" et les auteurs du papier [84] expliquent comment choisir les paramètres σ et ξ. DENCLUE présente un bon passage à l'échelle, puisqu'à sa phase initiale, il construit une carte de cubes hyperrectangles d'une longueur d'arrête 2σ. Pour cette raison, l'algorithme peut-être classé comme une méthode à base de grille. Tandis qu'aucun algorithme de clustering n'a une complexite inferieure à $O(N)$, le temps d'execution de DENCLUE se raproche de N sous-linéaire. L'explication est que quoique tous les points sont explorés, la majeure partie de l'analyse (dans la phase de clustering) concerne seulement les points des zones densement peuplés. Dans le tableau 1.3 nous avons résumé et comparé les caractéristiques générales des plus représentatifs algorithmes décrits dans cette section.

Méthode	***Type des données***	***Complexité***	***Géométrie***	***Resistance aux outliers, bruit***	***Paramètres d'entrée***	***Résultats***
DBSCAN	Numérique	$O(N \log N)$	Formes arbitraires	Oui	Le rayon du cluster, le nombre minimal des objets	L'affectation des valeurs de données aux clusters
DENCLUE	Numérique	$O(N \log N)$	Formes arbitraires	Oui	Le rayon du cluster σ, le nombre minimal des objets ξ	L'affectation des valeurs de données aux clusters

TAB. 1.3 – Les caractéristiques principales des algorithmes à base de densité (où N est le nombre des observations considérées).

1.5 Les méthodes à base de grille

Dans la section précédente, nous avons utilisé des concepts majeurs tels que la densité, les limites et la connectivité. Une autre manière de traiter ces concepts est d'extraire la topologie de l'espace des attributs sous-jacent. Pour limiter les combinaisons (possibilités) des recherches, on considère des segments multirectangulaires. On rappelle qu'un segment (appelé aussi *cube, cellule, région*) est un produit cartésien direct des sous-rangs de chaque variable (adjacent dans le cas des variables numériques). Le segment élémentaire qui correspond à des sous-rangs d'une seule valeur ou un seul histogramme est appelé **unité**.

Le partitionnement des données est induit par l'appartenance des points dans des segments qui résultent de l'espace de partitionnement, alors que l'espace de partitionnement est basé sur les caractéristiques de la grille extraites des données d'entrée. Un avantage de ce traitement indirect est que l'accumulation des données sous forme de grille rend les techniques à base de grille indépendante par rapport à l'ordre (rangement) des données. En revanche, les méthodes par réaffectation et les méthodes incrémentales sont très sensibles à l'ordre des données. Tandis que les méthodes à base de densité fonctionnent mieux sur des variables numériques, les méthodes à base de grille fonctionnent avec des variables de différents types. Dans une certaine mesure, les méthodes à base de grille reflètent un point de vue technique. Cette famille d'approches est éclectique : elle contient les deux types d'algorithmes : hiérarchiques et de partitionnement. L'algorithme DENCLUE de la section précédente (1.4.2) utilise les grilles pour l'étape initiale. Dans cette section nous allons étudier des algorithmes qui utilisent la technique à base de grille comme une heuristique principale.

La classification BANG [139] améliorent l'algorithme hiérarchique similaire GRID-CLUST [138]. Les segments à base de grille sont utilisés pour résumer les données. Les segments sont gardés dans une structure spéciale BANG qui est un répertoire de grille qui contient différentes échelles. Si une face commune a une dimension maximale, ils sont appelés les plus proches voisins. Plus généralement, on peut définir des voisins de degré entre 0 et d-1. La densité d'un segment peut être définie comme un taux du nombre de ses points et son volume. A partir du répertoire grille un dendogramme est directement calculé.

L'algorithme STING(STatistical INformation Grid-based method) [154] opère avec des attributs numériques (des données spatiales) et est désigné pour faciliter des " régions

orientées ". En agissant de telle manière, STING construit les résumés des données d'une manière similaire à BIRCH. WaveCluster [142] est basé sur l'idée du traitement du signal. Il utilise les transformations des ondelettes pour filtrer les données. Les parties d'une haute fréquence du signal correspondent aux frontières, tandis que les parties d'une grande amplitude avec une fréquence basse correspondent à l'intérieur des classes. La transformation par des ondelettes nous fournit des filtres utiles. Par exemple, les filtres sous forme de chapeau forcent les zones denses de servir comme des attracteurs et simultanément suppriment des zones aux frontières moins denses. Le fait de revenir de l'espace du signal à l'espace des attributs rend les classes plus fines et élimine les données atypiques. La complexité de l'algorithme est en $O(N)$ pour des petites dimensions, et grandit d'une manière exponentielle avec la dimension. L'algorithme WaveCluster est aussi connu pour d'autres propriétés remarquables :

- Une bonne qualité des classes
- Il manipule avec succès les données atypiques
- Abilité de traiter des données spatiales de dimensions relativement grandes
- Une complexité en $O(N)$

La hiérarchie des grilles nous permet de définir le concept de HFD (Hausdorf Fractal Dimension). Il est fondamental pour la classification fractale FC (Fractal Clustering) pour des attributs numériques, qui traitent plusieurs couches de grilles (la cardinalité de chaque dimension est multipliée par 4 avec chaque couche) [12]. Quoique pour économiser la mémoire, seulement les cellules occupées sont utilisées, l'utilisation de la mémoire reste un problème ouvert. FC commence avec l'initialisation des k classes. Par la suite, FC examine toutes les données incrémentalement. Il rajoute aux classes une observation qui arrive avec une certaine augmentation de HFD. Si la plus petite augmentation dépasse un seuil τ, un point est déclaré comme donnée atypique, sinon l'observation est affecté d'une telle manière que le HFD soit minimum. L'algorithme FC a des propriétés intéressantes comme :

- Abilité de trouver des classes de formes irrégulières
- Une complexité de $O(N)$
- Une structure incrémentale
- Prêt pour les affectations en ligne

Il a aussi quelques défauts :

- Une dépendance de l'ordre des données
- Une forte dépendance de l'initialisation des classes
- Dépendance des paramètres (le seuil utilisé pour l'initialisation et τ)

Dans le tableau 1.4 nous avons résumé et comparé les caractéristiques générales des plus représentatifs algorithmes décrits dans cette section.

Méthode	***Type des données***	***Complexité***	***Géométrie***	***Outliers***	***Paramètres d'entré***	***Résultats***
WaveCluster	Données spatiales	$O(N)$	Formes arbitraires	Oui	Wavelets, le nombre de cellules de la grille pour chaque dimension, le nombre d'applications de la transformation des ondelettes	Les objets classifiés en groupes
STING	Données spatiales	$O(K)$, où K est le nombre de cellules de la grille au plus bas niveau	Formes arbitraires	Oui	Le nombre d'objets dans une cellule	Les objets classifiés en groupes

TAB. 1.4 – Les caractéristiques principales des algorithmes à base de grille (où N est le nombre des observations considérées).

1.6 Les modèles auto-organisés

Dans cette section nous allons présenter les méthodes à base des cartes auto-organisatrices. On a choisi de les décrire plus en détail puisque les cartes auto-organisatrices sont à la base des méthodes que nous proposons dans cette thèse.

Les cartes auto-organisatrices peuvent traiter différents types de problèmes et ont des bonnes capacités d'apprentissage. Leurs points forts incluent : l'adaptation, facile à implémenter, la parallélisation, la rapidité et la flexibilité.

1.6.1 La descente du gradient et les modèles auto-organisés

Les réaffectations de type douce ("soft") ont du sens, si la fonction objective des k-moyennes est légèrement modifié pour intégrer (similaire à EM) “des erreurs floues ”.

$$E^{'}(C) = \sum_{i=1:N} \sum_{j=1:K} \pi_{ij}^2 \|\mathbf{x}_i - \mathbf{w}_j\|^2$$

Les probabilités exponentielles π sont définies en se basant sur les modèles gaussiennes. Cela rend la fonction objective différentiable par rapport aux moyennes et permet l'application de la méthode de la *descente du gradient*. Marroquin et Girosi [112] ont présenté une introduction détaillée par rapport à ce sujet dans le cadre de la *quantification vectorielle*. La méthode de la descente du gradient dans le cas de k-means est connue comme l'algorithme LKMA (Local K-Means Algorithm). A chaque itération t, on modifie les moyennes w_j^t dans la direction de la descente du gradient.

$$w_j^{t+1} = \mathbf{w}_j^t + a_t \sum_{i=1:N} (\mathbf{x}_i - \mathbf{w}_j^t)\pi_{ij}^2$$

ou

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^t + a_t(\mathbf{x}_i - \mathbf{w}_j^t)\pi_{ij}^2$$

Dans le deuxième cas une observation $\mathbf{x}$ est sélectionnée aléatoirement. Les scalaires a_t satisfont un comportement asymptotique monotone et convergent vers zéro, les coefficients w sont définis à l'aide de π [26]. De telles mises à jour sont utilisées dans le cadre des cartes auto-organisatrices, nommées SOM (Self-Organised Map) [99]. Nous allons décrire en détail cette méthode, car elle est à la base du travaille que nous proposons. La carte de Kohonen est une carte topologique auto-adaptative, qui cherche à partitionner l'ensemble des observations en groupements similaires. La structure peut être représentée comme un ensemble de cellules interconnectées et liées entre elles par une

structure de graphe non orienté, noté $\mathcal{C}$. Cette dernière définit une structure de voisinage entre les cellules (voir figure 1.4.). L'algorithme de Kohonen, permet de minimiser la fonction d'énergie suivante :

$$E_{SOM}(\chi, \mathcal{W}) = \sum_{i=1}^{N} \sum_{j=1}^{K} \mathcal{K}_{j,\chi(\mathbf{x}_i)} \|\mathbf{x}_i - \mathbf{w}_j\|^2 \tag{1.7}$$

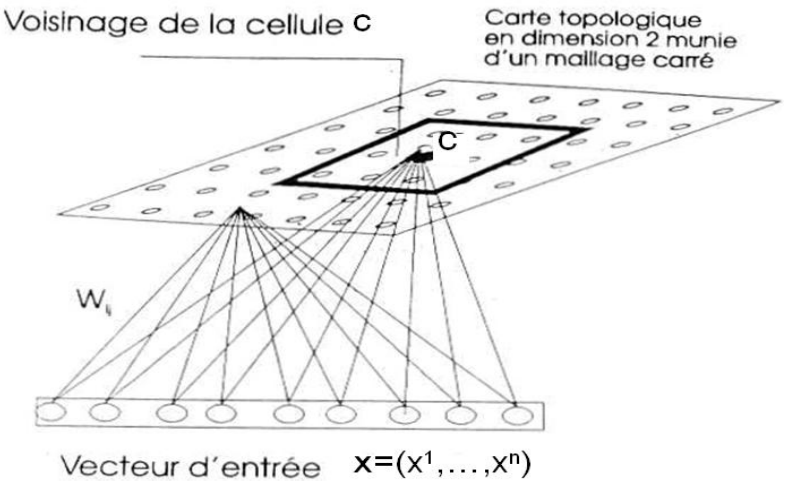


FIG. 1.4 – Carte topologique, constituée d'un treillis de cellules muni d'un voisinage et entièrement connecté

Les cellules de la carte sont donc réparties aux noeuds d'un maillage bidimensionnel, ces noeuds étant indexés par des nombres entiers qui définissent une relation de voisinage entre les cellules. La topologie de la carte est calculée (ou " apprise ") par un algorithme de gradient stochastique, appelé " algorithme de Kohonen ", auquel on fournit en entrée des données de dimension n (n pouvant être très supérieur à deux) à analyser.

Chaque cellule de la carte correspond alors à un " prototype " du jeu de données, c'est à dire un individu fictif représentatif d'un ensemble d'individus réels proches de lui-même (i.e. d'une classe d'individus réels). Une cellule de la carte est donc représentée par un vecteur de même dimension que les données, noté $\mathbf{w}$. Les composantes de ce vecteur sont souvent appelées les " poids " (notés $\mathbf{w}$ sur la figure 1.4) et sont également les coordonnées du prototype associé à la cellule dans l'espace multidimensionnel de départ. La propriété d'auto-organisation de la carte lui permet de passer d'un état désorganisé, suite à un positionnement aléatoire des prototypes à l'initialisation, à un

état organisé respectant la topologie des données.

Algorithme d'apprentissage de la carte :
En effet, l'algorithme de Kohonen poursuit simultanément deux buts :

- trouver les meilleurs prototypes (représentants) possible du jeu de données (ce qu'on appelle la " quantification vectorielle ")
- trouver une configuration des prototypes telle que deux prototypes proches dans l'espace des données soient associés à des cellules voisines sur la carte, cette proximité étant généralement au sens d'une métrique euclidienne, ou encore à ce que des cellules topologiquement proches sur la carte réagissent à des données d'entrée similaires.

Cet algorithme est de type compétitif : lors de la présentation d'un individu à la carte, les cellules entrent en compétition, de telle sorte qu'une seule d'entre elles, la " gagnante ", soit finalement active. Dans l'algorithme de Kohonen, le vainqueur est le neurone dont le prototype présente le moins de différence (souvent au sens d'une métrique euclidienne) avec l'individu présenté à la carte. Le principe de l'apprentissage compétitif consiste alors à récompenser le vainqueur, c'est-à-dire à rendre ce dernier encore plus sensible à une présentation ultérieure du même individu. Pour cela, on renforce les poids $\mathbf{w}$ avec les entrées. Les cellules se comportent à terme comme de véritables détecteurs de traits caractéristiques présents au sein des données d'entrée, chaque cellule se spécialisant dans la reconnaissance d'un trait particulier.

La cellule ayant remporté la compétition détermine le centre d'une zone de la carte appelée voisinage, zone dont l'étendue (rayon) varie au cours de l'apprentissage. La phase suivante, dite de mise à jour (ou adaptation), modifie la position des prototypes de façon à les rapprocher de l'individu présenté. Les prototypes sont d'autant plus rapprochés de l'individu en question qu'ils sont proches sur la carte de la cellule gagnante. En résumé, les étapes de l'algorithme de Kohonen sont les suivantes :

1) Initialisation des prototypes,
2) Sélection d'un individu,
3) Determination de la cellule gagnante pour cet individu (phase de "compétition"),
4) Modification de la totalité des prototypes de la carte (phase d' "adaptation"),
5) Reprise à l'étape 2 si condition d'arrêt non remplie.

A la fin de l'apprentissage, la carte est utilisée on l'utilise pour classer le jeu de données, en calculant simplement les distances euclidiennes entre le vecteur à n dimensions

caractérisant une donnée (individu) et les vecteurs (également à n dimensions) représentant les cellules de la carte. L'individu est alors attribué à la cellule dont il est le plus proche au sens de cette distance. On effectue cette opération pour tous les individus, et on obtient ainsi un classement du jeu de données. De plus, la carte permet de visualiser la proximité entre les classes obtenues (et donc aussi, entre les individus de ces classes).

1.6.2 Neuronale Gas (NGAS)

Des modèles dont l'idée centrale est très proche de l'algorithme de Kohonen ont été proposés par Martinetz [113] et repris dans différents travaux de recherche [67, 65, 66, 127] pour le traitement de données quantitatives. La différence entre les deux méthodes réside dans la définition des liens de voisinage. Pour les cartes de Kohonen, on définit a priori la topologie du réseau (par exemple une grille) ; dans le réseau Neural Gas, les liens entre les cellules ne sont pas fixés a priori mais se créent d'une manière dynamique durant l'apprentissage. Pendant cette phase, les référents non seulement se positionnent dans l'espace des données, mais établissent également des connexions voisines ou suppriment des connexions "parasites" (par exemple des cellules qui ne sont pas actives), donc ils constituent en même temps la structure graphique de la carte. Pour celà deux variables sont introduites pour définir les connexions : la première variable C_{ij} indique s'il existe une connexion de la cellule c_i vers la cellule c_j (prend la valeur 0 ou 1), la deuxième variable notée G_{ij} indique l'âge de la connexion et constitue un indicateur pour détecter celles qui sont jugées "parasites".

Le choix de la cellule gagnante par la fonction d'affectation χ ainsi que la loi d'adaptation des poids sont similaires à l'algorithme de Kohonen. La différence est que dans Neural Gas, on utilise une fonction d'ordre, qui n'est plus liée à la topologie de la carte (voisinage carré, hexagonal...) mais qui se définit à partir des plus proches voisins au sens de la distance euclidienne. Pour chaque observation $\mathbf{x}$, une liste ordonnée, au sens de cette distance, des référents $\mathbf{w}_c$ les plus proches est établie. La fonction $Cl_{\mathbf{x}}(\mathbf{w}_c)$ indique le classement ou l'ordre du référent $\mathbf{w}_c$ par rapport à l'observation $\mathbf{x}$ dans cette liste. Les poids des référents sont adaptés en fonction de cet ordre et à l'aide des fonctions noyaux.

L'algorithme NGAS

1) **Initialisation**

- $t = 0$, choisir un système de référent initial $\mathcal{W}^0$, la température T définissant le voisinage, le nombre d'itérations N_{iter} et le seuil g qui définit l'âge maximal d'une connexion.

2) **Etape itérative :** à l'itération t, on suppose connu $\mathcal{W}^{t-1}$ et la fonction d'affectation χ^{t-1} calculée à l'itération $t-1$.

- **Choisir** un exemple $\mathbf{x}$ de la base.

- **Phase d'affectation :** Prendre comme nouvelle fonction d'affectation χ^t. Pour chaque référent $\mathbf{w}_c^t$ déterminer son ordre $Cl(w_c^t)$ par rapport à l'exemple courant $\mathbf{x}$.

$$||\mathbf{x} - \mathbf{w}_{c_1}|| < ||\mathbf{x} - \mathbf{w}_{c_2}||... < ||\mathbf{x} - \mathbf{w}_{c_i}|| < ... < ||\mathbf{x} - \mathbf{w}_{c_p}||$$

avec l'ordre $i = Cl_{\mathbf{x}}(\mathbf{w}_c^t)$ et $c = 1..p$

- **Phase d'adaptation** chaque référent $\mathbf{w}_c^t$ est calculé selon l'expression :

$$\mathbf{w}_c^t = \mathbf{w}_{\chi(\mathbf{x}_i)}^{t-1} - \mu^t K^T(Cl_{\mathbf{x}}(\mathbf{w}_c^t))||\mathbf{x}_i - \mathbf{w}_c^{t-1}|| \tag{1.8}$$

Mise à jour des connexions

- Si $C_{i_1 i_2} = 0$ alors $C_{i_1 i_2} = 1$ (ajout d'une connexion) et $G_{i_1 i_2} = 1$. sinon $G_{i_1 i_2} = 1$.

- Pour toute connexion adjacente à i_1 $[i_1 j]$ ($C_{i_1 j} = 1$), si $G_{i_1 j} > g$ (g est un seuil prédéfini) alors supprimer la connexion : $C_{i_1 j} = 0$ et $G_{i_1 j} = 0$, sinon incrémenter l'âge de la connexion $G_{i_1 j} = G_{i_1 j} + 1$.

Répéter l'étape itérative jusqu'à atteindre N_{iter}.

Il y a donc une dispersion d'un certain nombre de cellules dans l'espace de données puis arrangement de ces cellules suivant une règle de plus proches voisins ; on adapte fortement le neurone gagnant, puis un peu le second, un peu moins le troisième, etc. Les unités gagnantes sont adaptées selon leur ordre dans la séquence. La fonction noyau $\mathcal{K}^T(.)$ est utilisée pour définir un ordre d'influence au lieu d'un voisinage d'influence ; souvent la taille de cet ordre décroît au cours des itérations. De plus, au cours des itérations des connexions s'établissent entre les neurones les plus actifs et d'autres

connexions qui sont jugées "parasites" disparaissent.

1.6.3 Generative Topographic Mapping (GTM)

"Generative Topographic Mapping" [22] est un modèle de cartes topologiques, qui est construit par estimation de fonctions densités en supposant l'existence de variable cachée $\mathbf{c}$ (non observée) soujacente à toute observation $\mathbf{x}$ réellement observée. Il s'agit donc de l'estimation de la fonction densité par la méthode des variables cachées. Cette méthode suppose donc le choix d'un modèle de la fonction densité jointe $p(\mathbf{x}, \mathbf{c}; W)$ où W est un ensemble de paramètres à estimer. La détermination de la fonction densité de la variable observée $\mathbf{x}$ se construit par marginalisation de la fonction densité jointe : $p(\mathbf{x}; W) = \int p(\mathbf{x}, \mathbf{c}; W)dc$. Le choix du modèle de la fonction densité peut se faire par l'intermédiaire de la décomposition : $p(\mathbf{x}, \mathbf{c}) = p(\mathbf{x}/\mathbf{c})p(\mathbf{c})$. Il s'agit alors de définir chacun de ces deux termes. Dans le cas du modèle GTM, on suppose que :

- La variable $\mathbf{c}$ est une variable discrète ayant K valeurs possibles $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K\}$ et que ces valeurs correspondent à des points situés dans le plan $\Re^2$ et forment les K sommets d'une grille régulière.

- La variable $\mathbf{c}$ est uniformément distribuée : $p(\mathbf{c}) = \frac{1}{K} \sum_{k=1.K} \delta(\mathbf{c} - \mathbf{c}_k)$.

Sous ces deux hypothèses nous pouvons écrire

$$p(\mathbf{x}; W) = \frac{1}{K} \sum_{k=1.K} f_k(\mathbf{x}; W)$$

où
$f_k(\mathbf{x}; W)$ représente la fonction densité de la variable conditionnelle $\mathbf{x}/\mathbf{c}_k$.

Une première version de GTM a été proposée par Bishop et al [22] dans le cas des données numériques . Elle propose, de modéliser les fonctions densités $f_k(\mathbf{x}; W)$ de la manière suivante :

- La fonction f_k est une fonction densité normale sphérique (dont la matrice de variance-covariance est σI).

- La moyenne de cette fonction densité est déterminée par : $\mathbf{t}_k = y(\mathbf{c}_k; W)$ où y est une fonction de $\Re^2$ dans $\Re^n$ (espace des données). Dans le modèle GTM y est

choisie comme une somme linéaire généralisée $y(\mathbf{c}; W) = W\Phi(\mathbf{c})$ et $\mathbf{\Phi}$ est formé de M fonctions de bases Φ_j. Ces fonctions Φ_j sont continues et prédéfinies et souvent des fonctions radiales de bases ayant une variance constante. Ainsi, les paramètres du modèle sont les coefficients de la matrice W. L'algorithme d'estimation des paramètres W découle de l'application de l'algorithme EM sur la fonction de vraisemblance des observations.

Avec le modèle GTM, l'ordre topologique est introduit par la fonction y qui est une fonction continue de $\Re^2$ dans $\Re^n$. La continuité de cette fonction permet d'associer à deux points proches dans l'espace euclidien $\Re^2$ deux images voisines dans l'espace euclidien $\Re^n$. Ainsi, la conservation de la topologie est assurée par le biais de la fonction y et le fait que l'on dispose d'une grille régulière dans $\Re^2$. Par contre le modéle GTM introduit une représentation interne des points de la grille par l'intermédiaire des M fonctions Φ_j. Cette représentation non linéaire des points de la grille dans un espace de dimension M permet de capter des corrélations, à travers des variables cachées sous-jacentes. De ce point de vue, les points de la grille permettent de générer l'échantillon des points de l'espace $\Re^M$.

Le modèle GTM a été étendu [72, 91] en considérant des fonctions f_k du type exponentielles définies par :

$$f_k(\mathbf{x}, W) = \exp\{W\Phi(\mathbf{c}_k)\mathbf{x} - G(W\Phi(c_k))\}p_0(\mathbf{x})$$

Or, il est connu que cette famille de fonctions densités contient les fonctions gaussiennes, les lois binomiales, les lois multinomiales ainsi que la loi de Poisson [13]. Ce qui permet une extension de ce modèle aux variables binaires et catégorielles [72, 91].

1.7 L'évaluation des résultats de la classification

Dans les sections précédentes, on a discuté plusieurs approches dont le but final était de fournir un bon partitionnement. Chaque algorithme peut diviser les données, mais différents algorithmes ou paramètres d'entrée produisent différents résultats, ou révèlent différentes structures des clusters. Ainsi le problème d'évaluer objectivement et quantitativement les clusters qui résultent tient de la *validation de la classification.*

C'est à ce problème qu'on va s'intéresser dans les sections qui suivent, et plus spécifiquement nous allons parler de la qualité de la classification, quel est le bon nombre des classes, etc.

1.7.1 Qualité de la classification

Le processus de classification dans le domaine du data mining commence avec l'estimation (appréciation) si une tendance de classification a lieu ou non, et, respectivement, inclue une bonne sélection des variables, et dans la plupart des cas la construction de nouveaux attributs ("features"). Il finit avec la *validation* et l'*évaluation* du système des classes qui résultent. Les résultats peuvent être évalués par un expert, ou par une procédure automatique particulière. Généralement, le premier type d'évaluation est lié à deux caractéristiques : (1) l'interprétabilité des classes, (2) la visualisation des classes. L'interprétabilité dépend des techniques utilisées. Les algorithmes à base de modèles probabilistes et les algorithmes conceptuels, comme COBWEB, ont des meilleurs résultats de ce point de vue. Les méthodes k-means et k-médoïdes génèrent des classes qui sont interprétées autour des centroïdes ou des médoïdes, et par conséquent, ont aussi des bons résultats. La revue [88] décrit bien le sujet de la validation de la classification, tandis que des discussions sur la visualisation des classes et des références reliées peuvent être trouvées dans [92].

En ce qui concerne les procédures automatiques, quand deux partitions sont construites (avec le même ou différent nombre de sous-ensembles de k), la première chose à faire est de les comparer. Parfois, l'étiquette de la classe actuelle s d'une partition est connue, mais le processus de classification est censé générer une autre étiquette j. La situation est similaire au test d'un classifieur dans le domaine de la fouille de données prédictive quand la cible actuelle est connue. La comparaison des étiquettes s et j est equivalente à calculer une erreur, la matrice de confusion, etc., en fouille de données prédictive. Il y a un critère simple **Rand** qui sert à cela [135]. Le calcul du critère de Rand implique des paires de points qui ont été affectés aux mêmes ou aux différentes classes. Il a une complexité de $O(N^2)$ qui n'est pas toujours réalisable. **L'entropy conditionnelle** d'une étiquette connue s ayant le partitionnement des classes [39] :

$$H(S|J) = -\sum_j p_j \sum_s p_{s|j} log(p_{s|j})$$

est une autre mesure utilisée. Ici $p_j, p_{s|j}$ sont les probabilités a priori de la classe j, et les probabilités conditionnelles de s sachant j. Elle a une complexité $O(N)$. Il existe

d'autres mesures qui sont utilisées, par exemple, la F-mesure [102]. Nous, par la suite, pour les approches qu'on propose on les comparent suelement utilisant la pureté.

1.7.2 Quel est le bon nombre des classes ?

Dans beaucoup de méthodes le nombre k des classes qu'on doit obtenir représente un paramètre d'entrée donné par l'utilisateur. En lançant un algorithme plusieurs fois on obtient une séquence de systèmes de classes. Chaque système fournit des classes plus granulaires et moins séparées. Dans le cas des k-means, la fonction objective est monotone décroissante. Par conséquent, la réponse à la question quel système est préférable n'est pas triviale.

Il existe plusieurs critères qui sont introduits pour trouver un k optimal. Parmis les premières publications sur la séparation des classes pour différents k on a [52, 118]. Par exemple, une distance entre n'importe quels deux centroïdes (médoïds) normalisée par rapport à l'écart de la classe (écart-type) et pondérée (avec le poids des classes) représente un choix raisonnable d'un *coefficient de séparation*. Ce coefficient a une complexité très basse $O(k^2)$. Une autre mesure de séparation populaire est le coefficient de Silhouette [96] avec la complexité $O(N^2)$. Si on considère la distance moyenne entre l'observation $\mathbf{x}$ de la classe c et d'autres observations dans la classe c et si on la compare avec la distance moyenne pour le cluster le plus adéquat G autre que c :

$$a(\mathbf{x}) = \frac{1}{|c|-1} \sum_{y \in c, y \neq x} d(\mathbf{x}, \mathbf{y}), b(\mathbf{x}) = \min_{G \neq c} \frac{1}{|G|} \sum_{y \in G} d(\mathbf{x}, \mathbf{y})$$

Le coefficient de Silhouette de $\mathbf{x}$ est $s(\mathbf{x}) = (b(\mathbf{x}) - a(\mathbf{x}))/max\{a(\mathbf{x}), b(\mathbf{x})\}$, les valeurs proche de $+1$ correspondent au cas parfait et les valeurs en dessous de 0 à un mauvais choix de classification. La moyenne globale des $s(\mathbf{x})$ individuels donne une bonne indication sur les classes obtenues.

Une autre manière pour la séparation est d'utiliser l'affectation douce (ou fuzzy). Il a une complexité intermediaire $O(kN)$. On rappelle que l'affectation d'une observation à une classe particulière peut souvent impliquer certains cas arbitraires. En fonction de l'adéquation d'une observation à une classe C, différentes probabilités ou poids $w(\mathbf{x}, c)$ peuvent être introduites tel qu'une affectation dure (hard) est définie de la manière suivante :

$$c(\mathbf{x}) = \arg\min_{c} w(\mathbf{x}, c)$$

Un coefficient de partitionnement [20] est egal à la somme des carrés des poids :

$$W = \frac{1}{N} \sum_{\mathbf{x} \in \mathcal{A}} w(\mathbf{x}, c(\mathbf{x}))^2$$

Chacune des mesures discutées peut être affichée comme une fonction de k et le graphe peut être utilisé pour choisir le meilleur k.

Un bon formalisme probabiliste des modèles des mélanges, discuté dans la sous-section 1.3.1, nous permet de voir le choix du k optimal comme un problème de choix du meilleur modèle pour les données. La question est : si on rajoute de nouveaux paramètres, obtient-on un meilleur modèle ? Dans l'apprentissage Bayesian (par exemple, AUTOCLASS [37]) la vraisemblance du modèle est directement affectée par la complexité du modèle (le nombre des paramètres est proportionnel à k). Plusieurs critères ont été introduits :

- Le critère MDL (Minimum Description Length) [136, 140, 87]
- Le critère MML (Minimum Message Length) [153, 152]
- Le critère BIC (Bayesian Information Criterion) [140, 64]
- Le critère AIC (Akaike's Information Criterion) [32]
- Le critère ICOMP (Non-coding Information Theoretic Criterion)[31]
- Le critère AWE (Approximate Weight of Evidence)[11]
- Bayes Factors [95]

Tous ces critères sont exprimés par la combinaison du log-vraisemblance L, nombre des classes k, le nombre des paramètres par classe, le nombre total des paramètres estimés p, et différents critères sur la matrice d'information de Fisher. Par exemple :

$$MDL(k) = -L + p/2 - \log(p), k_{best} = \min \arg MDL(k) \tag{1.9}$$

$$BIC(k) = L - \frac{p}{2} log(N), k_{best} = \max \arg BIC(k)$$

Des détails et des discussions supplémentaires peuvent être trouvées dans [25, 128, 64]. Quelques exemples : MCLUST et X-means utilisent le critère BIC, SNOB utilise le critère MML, CLIQUE utilise MDL. Par exemple, différents niveaux de BIC sont utilisés pour une preuve faible, positive et très forte pour la faveur d'un système de classification contre un autre [63]. Smyth [144] a proposé une technique de cross-validation de la vraisemblance pour la détermination des meilleurs k.

1.8 Conclusion

La classification automatique est l'une des tâches majeures dans différents domaines de recherche. Cependant, on peut la trouver sous différents noms et dans différents contextes : comme apprentissage non-supervisé dans le domaine de la reconnaissance de formes,taxonomie en biologie,analyse typologique dans les sciences sociales et partition dans la théorie des graphes. Le but de la classification automatique est d'identifier et extraire des groupes significatifs dans les données. Pour ce faire, il existe plusieurs méthodes.

Dans ce chapitre on a structurées les approches qui existent dans différentes catégories (c'est-à-dire hiérarchique, par partitionnement, à base de densité, à base de grille) et pour chaque type d'approche on a présenté un tableau qui résume les principales caractéristiques qui les définissent, mais dans le cadre de cette thèse nous nous intéressons plus particulièrement aux algorithmes à base de modèle des cartes auto-organisatrices. Un autre problème important que nous avons discuté dans le cadre de ce chapitre est l'évaluation des résultats de la classification. La plupart des algorithmes se basent sur certains critères pour définir les classes dans lesquelles les données vont être divisées. Puisque la classification automatique est une méthode non-supervisé et il n'y a pas d'indices a priori sur les classes obtenues, on a besoin de certains critères de validation des résultats. Ainsi, tous ces critères de validation nous permettent de justifier les résultats de la classification et d'assurer une bonne compréhension de la structure sous-jacente des données.

Chapitre 2

Classification probabiliste topologique de données binaires

2.1 Introduction

L'introduction d'un modèle qui permet d'exprimer la densité des observations en fonction de lois de paramètres amène tout naturellement à maximiser la vraisemblance de l'ensemble des observations et à déterminer par cette maximisation les paramètres optimaux du mélange représenté par la carte. Plusieurs méthodes peuvent êtres utilisées pour atteindre l'optimum : formalisme des nuées dynamiques, algorithme du gradient, algorithme d'Estimation de Maximisation. Chacun permet de proposer un algorithme dont la forme est semblable à ceux que nous avons présentés dans le premier chapitre.

Un algorithme utilisant la modélisation par des modèles de mélange a été proposé pour les données réelles, $\Re^n$. Cet algorithme appelé PrSOM (PRobabilistic Self-Organizing Map (PrSOM)), [5, 149], suppose que la distribution de probabilité $p(\mathbf{x}/c)$ associée à chaque cellule de la carte, prend une forme analytique qui est représentée par une loi Gaussienne sphérique. Nous allons noter par la suite par $\mathcal{A} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ l'ensemble des observations $\mathbf{x}$ constitué de n composantes $\mathbf{x} = (x^1, ..., x^n)$. Chaque fonction densité est définie par son vecteur moyen qui est le référent $\mathbf{w}_c = (w_c^1, ..., w_c^n)$ ainsi que l'écart type σ_c qui varie maintenant avec chaque cellule c, figure 2.1. Les mélanges de fonctions considérées sont donc des mélanges de fonctions Gaussiennes. Les paramètres à estimer dans ce cas représentent, $\theta = (\mathcal{W}, \Sigma)$, l'ensemble des référents $\mathcal{W}$ et l'ensemble des écarts-types $\Sigma = \{\sigma_c, c = 1..K\}$.

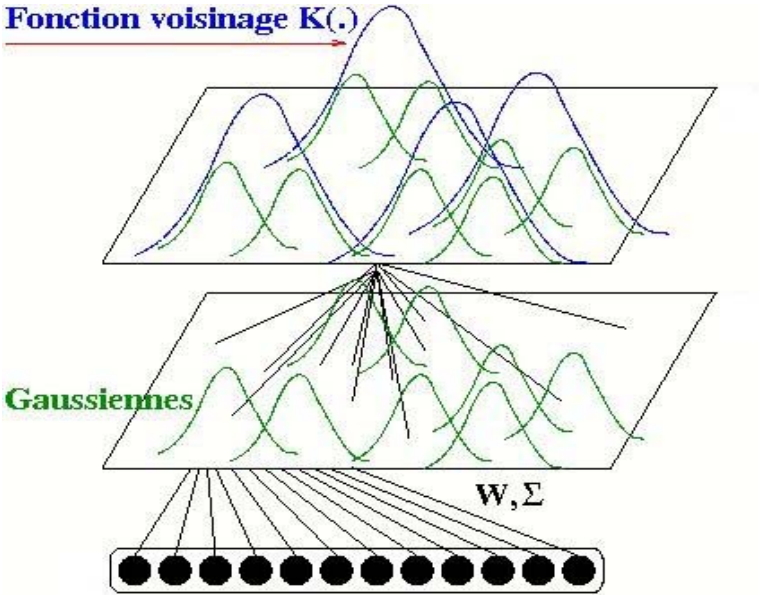


FIG. 2.1 – Modélisation de la carte auto-organisatrice sous forme d'un modèle de mélange Gaussien

Dans le cas du PrSOM, l'estimation se fait par maximisation de la vraisemblance classifiante à l'aide d'un algorithme de gradient. Nous ne présentons pas ici ce modèle, mais nous détaillerons par la suite de ce chapitre (section 2.3) l'algorithme EM et le modèle similaire à notre approche qui sera utilisée pour estimer les paramètres de la carte topologique probabiliste dédiée aux données qualitatives codées en binaire.

2.2 Modèle de mélange

Pour définir le modèle des cartes topologiques à base de modèle de mélange on associe à chaque cellule c de la carte $\mathcal{C}$ une fonction densité $\mathbf{f}_c(\mathbf{x}) = p(\mathbf{x}/\theta_c)$ dont les paramètres seront notés θ. Pour définir le mélange de densités des cartes topologiques, nous utiliserons le formalisme bayésien introduit par Luttrel [108]. Ce formalisme suppose que les observations $\mathbf{x}$ sont générées de la manière suivante : on commence par choisir une cellule c^* de $\mathcal{C}$ qui permet de déterminer un voisinage de cellule suivant la probabilité conditionnelle $p(c/c^*)$ qui est supposée connue. Ainsi l'observation $\mathbf{x}$ est générée suivant la probabilité $p(\mathbf{x}/c)$. Ce processus permet de modéliser les différentes étapes de propagation de l'information entre les différentes cellules $c \in \mathcal{C}$ et $c^* \in \mathcal{C}$.

La première étape de propagation attribue à une observation $\mathbf{x}$ une cellule c de $\mathcal{C}$ avec la probabilité $p(c/\mathbf{x})$. La seconde étape attribue à toute cellule c de $\mathcal{C}$ une cellule c^* de $\mathcal{C}$ avec la probabilité $p(c^*/c)$. Afin de simplifier le calcul des probabilités on suppose que le processus de propagation vérifie la propriété de Markov :

$$p(\mathbf{x}/c, c^*) = p(\mathbf{x}/c)$$

et

$$p(c^*/c, \mathbf{x}) = p(c^*/c)$$

Ce formalisme nous amène à définir le générateur des données $p(\mathbf{x})$ par un mélange de probabilités défini comme suit :

$$\begin{aligned} p(\mathbf{x}) &= \sum_{c,c^* \in C} p(c, c^*, \mathbf{x}) \\ &= \sum_{c,c^* \in C} p(\mathbf{x}/c)p(c/c^*)p(c^*) \\ &= \sum_{c^* \in \mathcal{C}} p(c^*)p_{c^*}(\mathbf{x}) \end{aligned} \tag{2.1}$$

avec

$$p_{c^*}(\mathbf{x}) = p(\mathbf{x}/c^*) = \sum_{c \in \mathcal{C}} p(c/c^*)p(\mathbf{x}/c). \tag{2.2}$$

Ainsi, $p(\mathbf{x})$ apparaît comme un mélange des probabilités $p_{c^*}(\mathbf{x})$. L'observation $\mathbf{x}$ s'obtient premièrement par la sélection de c^* puis de c de $\mathcal{C}$, ensuite par la sélection de $\mathbf{x}$ à l'intérieur du sous-échantillon avec la probabilité $p(\mathbf{x}/c)$.

Les coefficients du mélange sont les probabilités $p(c^*)$ *a priori* et les fonctions densités relatives à chaque élément du mélange qui sont données par $p_{c^*}(\mathbf{x})$ (eq.(2.2)). Ce formalisme montre qu'on peut calculer $p(\mathbf{x})$ à condition de connaître pour chaque cellule c la fonction de densité $p(\mathbf{x}/c)$ et la probabilité $p(c/c^*)$ d'activation de la cellule c connaissant c^*.

Afin d'introduire la notion de voisinage dans le formalisme probabiliste, nous supposons que chaque cellule c de la carte $\mathcal{C}$ est d'autant plus active qu'elle est proche de la cellule

choisie c^* (figure 2.2), (eq.(2.1), eq.(2.2)). Ceci nous permet de définir la probabilité $p(c/c^*)$ en fonction de la fonction de voisinage $\mathcal{K}^T(.)$:

$$p(c/c^*) = \frac{\mathcal{K}^T(\delta(c, c^*))}{T_{c^*}} \tag{2.3}$$

où $T_{c^*} = \sum_{r \in C} \mathcal{K}^T(\delta(r, c^*))$, est un terme normalisant pour obtenir des probabilités.

Pour définir complètement $p(\mathbf{x})$ il reste à définir les coefficients du mélange $p(c^*)$ et les paramètres de la densité $p(\mathbf{x}/c)$. Ce formalisme a déjà été utilisé dans [6] et a permis de définir le modèle PrSOM dédié aux données continues (section 2.1). Ce modèle qui généralise le modèle classique des cartes topologiques introduit par Kohonen permet d'obtenir une quantification de l'espace des données, mais aussi une estimation des densités locales.

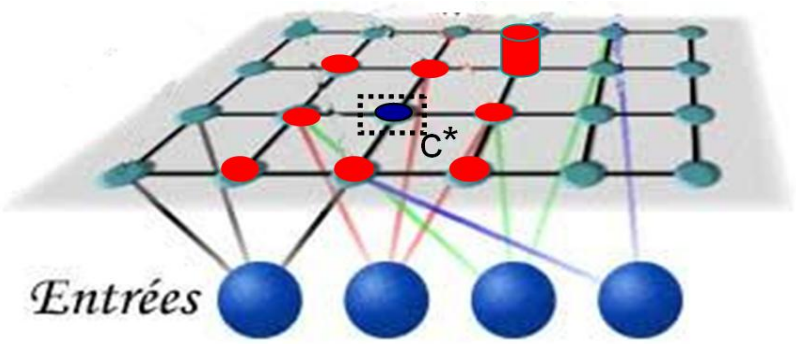


FIG. 2.2 – Une grille bi-dimensionnelle avec $\mathcal{C} = 4 \times 6$ cellules utilisées par le modèle générateur. La couleur rouge indique les cellules déterminées par la probabilité $p(c/c^*)$ et qui sont responsables de la génération de l'observation

2.3 Algorithme EM

L'algorithme EM [21, 46, 115] est un algorithme itératif qui permet de trouver un maximum local de la fonction de vraisemblance des observations, lorsque chaque observation contient une partie cachée (ou non observée). Ainsi, on suppose que chaque donnée est un couple de types $(\mathbf{x}, \mathbf{z})$ où $\mathbf{x}$ est sa partie observable et $\mathbf{z}$ sa partie cachée (non observable). Nous supposons connus d'une manière explicite la forme de la fonction densité jointe $p(\mathbf{x}, \mathbf{z}; \theta)$ où θ est l'ensemble des paramètres du modèle à estimer. On suppose

que l'on dispose d'une série de données indépendantes : $(\mathbf{x}_1, \mathbf{z}_1), (\mathbf{x}_2, \mathbf{z}_2), ..., (\mathbf{x}_N, \mathbf{z}_N)$, pour lesquelles $\mathbf{x}_i$ sont les parties qu'on a réellement observées et les $\mathbf{z}_i$ sont les parties cachées (donc inconnues).

Nous souhaitons par la suite maximiser le logarithme de la vraisemblance des parties, des données réellement observées $\mathcal{A} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ dont le logarithme est égal à :

$$\ln V(\mathcal{A}; \theta) = \ln V(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N; \theta) = \sum_{i=1}^{N} \ln p(\mathbf{x}_i; \theta) \tag{2.4}$$

où $p(\mathbf{x}; \theta)$ est la fonction densité de la partie observée $\mathbf{x}$. En pratique $p(\mathbf{x}; \theta)$ est calculable en marginalisant la fonction densité $p(\mathbf{x}, \mathbf{z}; \theta)$ ($p(\mathbf{x}; \theta) = \int p(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{z}$), ce qui donne souvent une fonction log-vraisemblance $\ln V(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N; \theta)$ qui n'est pas simple à optimiser.

L'algorithme EM proposé par Dempster et al [46] maximise l'expression (2.4) en utilisant le log-vraisemblance des données entières $\ln V(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N, \mathbf{z}_1, ..., \mathbf{z}_N; \theta)$. On désigne par $\mathbf{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_N\}$ l'ensemble des parties correspondantes et non observées. Chaque itération de l'algorithme EM comporte deux étapes :

- L'étape d'Estimation (Expectation step) ; dite aussi étape "E"
- L'étape de Maximisation (Maximization step) ; dite étape "M"

Ainsi à l'itération t ces deux étapes se présentent de la manière suivante :

- **Etape E (Expectation step)**
  On suppose à cette étape, que la fonction densité de la partie cachée conditionnée par la partie observée $(\mathbf{z}/\mathbf{x})$ correspond à la valeur du paramètre θ^{t-1} calculée à l'itération précédente (ou égale à l'initialisation θ^0 si $t = 1$) ; cette fonction densité s'écrit donc $(p(\mathbf{z}/\mathbf{x}, \theta^{t-1}))$. On calcule alors l'espérance :

$$\begin{aligned} Q(\theta, \theta^{t-1}) &= E\left[\ln V(\mathcal{A}, \mathbf{Z}/\theta)/\mathcal{A}, \theta^{t-1}\right] \\ &= \int \ln V(\mathcal{A}, \mathbf{Z}/\theta) p(\mathbf{Z}/\mathcal{A}, \theta^{t-1}) \\ &= \int \ln V(\mathcal{A}, \mathbf{Z}/\theta) \prod_{i=1}^{N} p(\mathbf{z}_i/\mathbf{x}_i, \theta^{t-1}) \; d\mathbf{z}_i \end{aligned} \tag{2.5}$$

Cette expression qui est parfois appelée "la vraisemblance relative" se justifie "intuitivement". En effet, étant donné qu'on ne connait pas les valeurs des variables cachées $\mathbf{z}_i$ associées aux observations $\mathbf{x}_i \in \mathcal{A}$, on calcule l'espérance du log-vraisemblance relativement aux variables cachées.

- **Etape de maximisation (Maximization step)**
  Ayant calculé $Q(\theta, \theta^{t-1})$ à l'étape E, il s'agit dans cette étape de maximiser cette expression par rapport à θ. On prend alors :

$$\theta^{'} = \arg\max_{\theta} Q(\theta, \theta^{t-1})$$

Il est démontré alors que chaque itération (E-M) fait croître la fonction log-vraisemblance (2.4) ($\ln V(\mathcal{A}, \theta^t) \geq \ln V(\mathcal{A}, \theta^{t-1})$) [46]. Ainsi, l'algorithme E-M se présente de la manière suivante :

---

1) **Initialisation** : Choisir des paramètres initiaux θ^0 et N_{iter} (le nombre d'itérations).
2) **Itération de Base** ($t \geq 1$)
   - Etape **E** : Estimer l'expression $Q(\theta, \theta^{t-1})$ définie par l'expression 2.5.
   - Etape **M** : Maximiser $Q(\theta, \theta^{t-1})$ par rapport à θ, prendre $\theta^t = \arg\max_\theta Q(\theta, \theta^{t-1})$
3) **Répéter** l'itération de base, jusqu'à stabilisation de θ^t ou jusqu'à $t \geq N_{iter}$

Remarque : L'algorithme EM est largement utilisé en classification pour bâtir de façon itérative, à partir d'un nombre d'observations données, des modèles de mélanges paramétriques.

2.4 Cartes auto-organisatrices binaires probabilistes

Les cartes topologiques probabilistes présentées sont des modèles de mélange dont il faut estimer les différents paramètres. Aux paragraphes qui suivent nous présentons un nouveau modèle de cartes topologiques adaptées aux données qualitatives codées en binaire et l'algorithme d'apprentissage associé qui permet d'estimer les différents paramètres de ce modèle. Nous parlerons de BeSOM (Bernoulli Self-Organising Map) pour

le modèle ou pour l'algorithme d'apprentissage permettant d'estimer les paramètres. L'algorithme BeSOM est une application directe de l'algorithme EM. Par la suite nous développerons trois versions de BeSOM (tableau 2.1). La première est développée sous l'hypothèse que le paramètre de probabilité ϵ dépend uniquement de la cellule ($\epsilon = \varepsilon_c$). La deuxième est le cas général où le paramètre de probabilité dépend de la cellule et de la variable $\epsilon_c = (\varepsilon_c^1, ..., \varepsilon_c^k, ..., \varepsilon_c^n)$. La troisième version estime un paramètre qui dépend seulement de la carte ($\epsilon = \varepsilon$).

BeSOM-ε_c	$\epsilon = \varepsilon_c$	$\mathbf{w}_c = (w_c^1, ..., w_c^k, ..., w_c^n)$
BeSOM-ϵ_c	$\epsilon_c = (\varepsilon_c^1, ..., \varepsilon_c^k, ..., \varepsilon_c^n)$	$\mathbf{w}_c = (w_c^1, ..., w_c^k, ..., w_c^n)$
BeSOM-ε	$\epsilon = \varepsilon$	$\mathbf{w}_c = (w_c^1, ..., w_c^k, ..., w_c^n)$

TAB. 2.1 – Les paramètres des trois versions BeSOM

Le formalisme des cartes topologiques probabilistes suppose que les données observées sont générées suivant la loi de mélange définie par les équations (eq.(2.1), eq.(2.2)). Les coefficients à estimer sont les paramètres de probabilités $p(c^*)$ et les paramètres des fonctions de densités relatives à chaque élément du mélange qui sont données par $p(\mathbf{x}/c)$. Dans la suite de ce chapitre, nous allons donner une forme particulière à ces fonctions qui permettent explicitement de prendre en compte les dépendances entre les modalités d'une variable qualitative distinctive.

2.4.1 Modèle de mélange de Bernoulli : BeSOM-ε_c

Dans le cas où les variables observées $\mathbf{x} = (x^1, x^j, ..., x^n)$ sont constituées de composantes x qualitatives codées en binaire, nous supposons que la probabilité $p(\mathbf{x}/c)$ se présente sous forme de loi de Bernouilli de paramètre ε et qui définit une variable binaire $x \in \{0, 1\}$ par : $p(x/\varepsilon) = \varepsilon^x(1-\varepsilon)^{1-x}$ [125]. D'une manière plus précise, on s'intéresse à la formule des lois de probabilités définie par :

$$\varepsilon^{|x-w|}(1-\varepsilon)^{1-|x-w|}$$

où $\varepsilon \in]0\ 1/2[$ et $w \in \beta = \{0, 1\}$.

Ainsi, x reproduit la valeur de w avec la probabilité $1-\varepsilon$ et elle est différente de x avec la probabilité ε.

Afin de définir $p(\mathbf{x}/\theta_c)$ (où $\mathbf{x} \in \mathcal{A} \subset \beta^n$ et $\theta_c = \{\varepsilon_c, w_c\}$), on fait par la suite l'hypothèse que les n composantes du vecteur aléatoire $\mathbf{x}$ sont indépendantes (non corrélées) et que la j^{ime} composante suit la loi :

$$p(x^j/\varepsilon_c, w_c^j) = \varepsilon_c^{|x^j - w_c^j|}(1-\varepsilon_c)^{1-|x^j - w_c^j|}$$

Ainsi, la loi de probabilité qui définit le vecteur aléatoire $\mathbf{x} = (x^1, x^2, ..., x^n)$ s'écrit :

$$p(\mathbf{x}/\theta_c) = p(\mathbf{x}/\mathbf{w}_c, \varepsilon_c) = \prod_{j=1}^{n} \varepsilon_c^{|x^j - w_c^j|}(1-\varepsilon_c)^{1-|x^j - w_c^j|}$$

où $\mathbf{w}_c = (w_c^1, w_c^2, ..., w_c^n) \in \beta^n$.

Cette formulation suppose que les différentes composantes de $\mathbf{x}$ suivent une loi de Bernoulli de même paramètre ε_c et que la j^{ime} composante reproduit la valeur booléenne qui lui est associée w_c^j avec la probabilité $1-\varepsilon_c$. De ce qui suit, on a la formule suivante :

$$p(\mathbf{x}/\mathbf{w}_c, \varepsilon_c) = \varepsilon_c^{\mathcal{H}(\mathbf{x},\mathbf{w}_c)}(1-\varepsilon_c)^{n-\mathcal{H}(\mathbf{x},\mathbf{w}_c)} \tag{2.6}$$

Où $\mathcal{H}$ indique la distance de Hamming qui calcule le nombre de différences entre les deux vecteurs binaires $\mathbf{x} \in \beta^n$ et $\mathbf{w}_c \in \beta^n$ (Annexe A).

Cette formulation qui a été proposée par Celeux et Govaert dans [35, 125], suppose que la loi de probabilité génère des observations $\mathbf{x}$ de β^n autour d'un vecteur fixe $\mathbf{w}_c$ de telle façon que tous les composants sont indépendants et non corrélés et chaque composante de $\mathbf{x}$ reproduit la composante correspondante de $\mathbf{w}_c$ avec la même probabilité $(1-\varepsilon_c)$. Ainsi, cette formulation introduit la notion de vecteur référent puisqu'elle suppose que la loi de probabilité considérée $p(\mathbf{x}/\mathbf{w}_c, \varepsilon_c)$ génère des données plus ou moins similaires à $\mathbf{w}_c$.

L'ensemble des paramètres permettant de définir la loi de Bernoulli d'une cellule c de la carte $\mathcal{C}$, est constitué de l'union de tous les couples $\theta_c = (\mathbf{w}_c, \varepsilon_c)$ qui constitue les paramètres de la loi de Bernoulli génératrice au niveau de chaque cellule c, $\theta^C = \cup_{c=1}^{K} \theta_c$.

Ainsi, les paramètres $\theta = \theta^C \cup \theta^{C^*}$ qui permettent de définir le modèle probabiliste de la carte topologique de taille p sont constitués par :

– l'ensemble des paramètres des loi de Bernoulli :

$$\theta^C = \cup_{c=1}^{K} \theta_c, \text{ où } \theta_c = (\varepsilon_c, w_c)$$

– l'ensemble des probabilités a priori :

$$\theta^C = \{\theta_{c^*}, c^* = 1..K\}, \ \theta_{c^*} = p(c^*)$$

Le problème est donc maintenant de définir l'algorithme d'apprentissage du modèle BeSOM permettant d'estimer l'ensemble de ses paramètres.

2.4.2 Estimation des paramètres du modèle

Dans le cadre du modèle des cartes topologiques, l'usage de l'algorithme EM s'explique par l'existence d'une variable cachée notée $\mathbf{z}$, constituée par le couple de cellules c et c^*, $\mathbf{z} = (c, c^*)$, responsable de la génération d'une donnée observée $\mathbf{x}$. En effet, la variable cachée $\mathbf{z} = (c, c^*)$ apparaît lorsqu'on écrit :

$$p(\mathbf{x}) = \sum_{z \in \mathbf{Z}} p(\mathbf{x}, \mathbf{z}) = \sum_{c \in \mathcal{C}, c^* \in \mathcal{C}} p(\mathbf{x}/c) p(c/c^*) p(c^*)$$

car $p(\mathbf{x}, c, c^*) = p(c^*)p(c/c^*)p(\mathbf{x}/c)$

Ainsi, à chaque donnée réellement observée $\mathbf{x}$, il lui correspond une donnée catégorielle disjonctive non observée $\mathbf{z}$ qui appartient à $\mathcal{C} \times \mathcal{C}$. Si l'on code cette variable par le codage binaire disjonctif, on obtient un vecteur binaire $\mathbf{z}$ de dimension $K \times K$ dont les composantes $z_{(c,c^*)}$ sont définies par :

$$z_i^{(c,c^*)} = \begin{cases} 1 & \text{pour } \mathbf{z}_i = (c, c^*) \\ 0 & \text{sinon} \end{cases} \tag{2.7}$$

Avec cette notation, $p(\mathbf{x}, \mathbf{z})$ s'écrit de la manière suivante :

$$p(\mathbf{x}, z) = \prod_{c^* \in \mathcal{C}} \prod_{c \in \mathcal{C}} \left[p(c^*) \frac{\mathcal{K}^T(\delta(c^*, c))}{T_{c^*}} p(\mathbf{x}/c) \right]^{z_{(c,c^*)}}$$

Par la suite, à chaque donnée réellement observée $\mathbf{x}_i$ de l'ensemble d'apprentissage $\mathcal{A}$, on suppose qu'il lui correspond une donnée non observée $\mathbf{z}_i \in \mathcal{C} \times \mathcal{C}$ qui sera codée par le vecteur binaire appartenant à $\{0,1\}^{K \times K}$ dont toutes les composantes sont nulles sauf la composante d'indice $z^i_{(c,c^*)}$ qui vaut 1. On a donc :

$$p(\mathbf{x}_i, z_i) = \prod_{c^* \in \mathcal{C}} \prod_{c \in \mathcal{C}} \left[p(c^*) \frac{\mathcal{K}^T(\delta(c^*, c))}{T_{c^*}} p(\mathbf{x}_i/c) \right]^{z^i_{(c,c^*)}} \tag{2.8}$$

Ainsi la vraisemblance des données s'écrit par :

$$V^T(\mathcal{A}, \mathbf{Z}; \theta) = \prod_{i=1}^{N} \prod_{c^* \in \mathcal{C}} \prod_{c \in \mathcal{C}} \left[\theta_{c^*} \frac{\mathcal{K}^T(\delta(c^*, c))}{T_{c^*}} p(\mathbf{x}_i/c) \right]^{z^i_{(c,c^*)}}$$

le log-vraisemblance s'écrit :

$$\ln V^T(\mathcal{A}, \mathbf{Z}; \theta) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c^* \in C} \sum_{c \in C} z^i_{(c,c^*)} \left[\ln(\theta_{c^*}) + \ln\left(\frac{\mathcal{K}^T(\delta(c^*, c))}{T_{c^*}} \right) + \ln(p(\mathbf{x}_i/c)) \right].$$

L'application de l'algorithme EM pour la maximisation de la vraisemblance des données observées nécessite d'une part l'estimation de $Q^T(\theta, \theta^t)$ (eq. 2.5) et, d'autre part, sa maximisation par rapport à l'ensemble des paramètres θ.

2.4.2.1 Estimation de $Q^T(\theta, \theta^t)$

On sait que :

$$Q^T(\theta, \theta^t) = E\left[\ln V^T(\mathcal{A}, \mathbf{Z}; \theta) / \mathcal{A}, \theta^t \right]$$

$$Q^T(\theta, \theta^t) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c^* \in C} \sum_{c \in C} E(z^i_{(c,c^*)}/\mathbf{x}_i, \theta^t) \left[\ln(\theta_{c^*}) + \ln\left(\frac{\mathcal{K}^T(\delta(c^*, c))}{T_{c^*}} \right) + \ln(p(\mathbf{x}_i/c)) \right]$$

où θ^t est l'ensemble des paramètres estimés à la t^{eme} itération de l'algorithme, et θ l'ensemble des paramètres recherchés. La variable $z^i_{(c,c^*)}$ étant une variable de Bernoulli, $E(z^i_{(c,c^*)}/\mathbf{x}_i, \theta^t) = p(z^i_{(c,c^*)} = 1/\mathbf{x}_i, \theta^t) = p(c, c^*/\mathbf{x}_i, \theta^t)$, tel que :

$$E(z^i_{(c,c^*)}/\mathbf{x}_i, \theta^t) = p(c, c^*/\mathbf{x}, \theta^t) = \frac{p(c^*)p(c/c^*)p(\mathbf{x}/c)}{\sum_{r\in\mathcal{C}} p(r)p_r(\mathbf{x})} \tag{2.9}$$

Ainsi, la fonction $Q^T(\theta, \theta^t)$ s'écrit :

$$\begin{aligned} Q^T(\theta, \theta^t) &= \sum_{\mathbf{x}_i\in\mathcal{A}} \sum_{c^*\in C} \sum_{c\in C} p(c, c^*/\mathbf{x}_i, \theta^t) \ln(p(\mathbf{x}_i/c)) \\ &+ \sum_{\mathbf{x}_i\in\mathcal{A}} \sum_{c^*\in C} \sum_{c\in C} p(c, c^*/\mathbf{x}_i, \theta^t) \ln(\theta_{c^*}) \\ &+ \sum_{\mathbf{x}_i\in\mathcal{A}} \sum_{c^*\in C} \sum_{c\in C} p(c, c^*/\mathbf{x}_i, \theta^t) \ln\left(\frac{\mathcal{K}^T(\delta(c^*, c))}{T_{c^*}}\right) \end{aligned} \tag{2.10}$$

On observe que la fonction $Q^T(\theta, \theta^t)$ se décompose en une somme de trois termes. Le premier terme ne dépend que des paramètres θ^C (paramètres de la loi de Bernoulli), il sera noté par la suite $Q^T_1(\theta^C, \theta^t)$. Le second terme ne dépend que de θ^{C^*} (l'ensemble des probabilité a priori), il sera noté par la suite $Q^T_2(\theta^{C^*}, \theta^t)$. Le troisième terme est constant par rapport à l'ensemble des paramètres θ et ne dépend que de θ^t, il est noté $Q^T_3(\theta^t)$. Ainsi,

$$Q^T(\theta, \theta^t) = Q^T_1(\theta^C, \theta^t) + Q^T_2(\theta^{C^*}, \theta^t) + Q^T_3(\theta^t)$$

2.4.2.2 Maximisation de la fonction $Q^T(\theta, \theta^t)$

Maximiser la fonction auxiliaire (eq.2.10) par rapport à θ revient à maximiser les deux premiers termes de cette fonction séparément $Q^T_2(\theta^{C^*}, \theta^t)$ par rapport à θ^{C^*} et la fonction $Q^T_1(\theta^C, \theta^t)$ par rapport θ^C, avec :

$$Q_1^T(\theta^C, \theta^t) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} \sum_{c^* \in C} p(c, c^*/\mathbf{x}_i, \theta^t) \ln(p(\mathbf{x}_i/c)) \tag{2.11}$$

et

$$Q_2^T(\theta^C, \theta^t) = \sum_{c^* \in C} \left[\sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} p(c, c^*/\mathbf{x}_i, \theta^t) \right] \ln(\theta_{c^*}) \tag{2.12}$$

1) **Maximisation de $Q_2^T(\theta^{C^*}, \theta^t)$**
Afin d'estimer les paramètre θ^{C^*}, il faut maintenant donner une forme explicite aux probabilités. On suppose pour le modèle BeSOM que les probabilités a priori θ_{c^*} s'expriment par une relation du type softmax :

$$\theta_{c^*} = \frac{e^{\lambda_{c^*}}}{\sum_{r \in C} e^{\lambda_r}} \tag{2.13}$$

On réécrit alors l'expression (2.12)

$$Q_2^T(\theta^{C^*}, \theta^t) = \sum_{c^* \in C} \left[\sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} p(c, c^*/\mathbf{x}_i, \theta^t) \right] \ln \left(\frac{e^{\lambda_{c^*}}}{\sum_{r \in C} e^{\lambda_r}} \right)$$

après développement on obtient les expressions suivantes :

$$\begin{aligned} Q_2^T(\theta^{C^*}, \theta^t) &= \sum_{c^* \in C} \lambda_{c^*} \left[\sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} p(c, c^*/\mathbf{x}_i, \theta^t) \right] \\ &\quad - \ln \left(\sum_{r \in C} e^{\lambda_r} \right) \left[\sum_{c^* \in C} \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} p(c, c^*/\mathbf{x}_i, \theta^t) \right] \end{aligned}$$

$$Q_2^T(\theta^{C^*}, \theta^t) = \sum_{c^* \in C} \lambda_{c^*} \left[\sum_{\mathbf{x}_i \in \mathcal{A}} p(c^*/\mathbf{x}_i, \theta^t) \right] - N \ln \left(\sum_{r \in C} e^{\lambda_r} \right)$$

La dérivée de la fonction $Q_2^T(\theta^{C^*}, \theta^t)$ par rapport au paramètre λ_{c^*} est égale à :

$$\frac{\partial Q_2^T(\theta^{C^*}, \theta^t)}{\partial \lambda_{c^*}} = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c^* \in C} p(c^*/\mathbf{x}_i, \theta^t) - N \frac{e^{\lambda_{c^*}}}{\sum_{r \in C} e^{\lambda_r}}$$

Utilisant l'expression (eq. 2.13) de la probabilité a priori,

$$\frac{\partial Q_2^T(\theta^{C^*}, \theta^t)}{\partial \lambda_{c^*}} = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c^* \in C} p(c^*/\mathbf{x}_i, \theta^t) - N\theta_{c^*}$$

Pour $\frac{\partial Q_2^T(\theta^{C^*}, \theta^t)}{\partial \lambda_{c^*}} = 0$, on obtient l'expression permettant d'estimer le paramètre θ_{c^*} :

$$\theta_{c^*} = \frac{\sum_{\mathbf{x}_i \in \mathcal{A}} p(c^*/\mathbf{x}_i, \theta^t)}{N} \tag{2.14}$$

avec $p(c^*/\mathbf{x}_i, \theta^t) = \sum_{c \in C} p(c, c^*/\mathbf{x}_i, \theta^t)$. L'expression (2.14) peut être écrite sous la forme suivante :

$$\theta_{c^*} = \frac{\sum_{\mathbf{x}_i \in \mathcal{A}} p(c^*/\mathbf{x}_i, \theta^t)}{\sum_{c^* \in C} \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} p(c, c^*/\mathbf{x}_i, \theta^t)} \tag{2.15}$$

2) **Maximisation de $Q_1^T(\theta^C, \theta^t)$**
On suppose que le paramètre θc constitue le paramètre de la loi de Bernoulli associée à la cellule c. D'autre part, étant donné qu'on a fait l'hypothèse de l'indépendance des composantes x^k de l'observation $\mathbf{x}$, on a $p(\mathbf{x}/c) = \prod_{k=1}^n p(x^k/c)$. D'où

$$Q_1^T(\theta^C, \theta^t) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} \sum_{c^* \in C} p(c, c^*/\mathbf{x}_i, \theta^t) \ln \left(\prod_{k=1}^n p(x_i^k/c) \right)$$

Ainsi,

$$\begin{aligned} Q_1^T(\theta^C, \theta^t) &= \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} p(c/\mathbf{x}_i, \theta^t) \ln \left(\prod_{k=1}^n p(x_i^k/c) \right) \\ &= \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} p(c/\mathbf{x}_i, \theta^t) \ln \left(\varepsilon_c^{\mathcal{H}(\mathbf{x}_i, \mathbf{w}_c)} (1 - \varepsilon_c)^{n - \mathcal{H}(\mathbf{x}_i, \mathbf{w}_c)} \right) \\ &= \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} p(c/\mathbf{x}_i, \theta^t) \left[\mathcal{H}(\mathbf{x}_i, \mathbf{w}_c) \ln(\varepsilon_c) n - \mathcal{H}(\mathbf{x}_i, \mathbf{w}_c) \ln(1 - \varepsilon_c) \right] \end{aligned}$$

$$\begin{aligned} Q_1^T(\theta^C, \theta^t) &= \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} p(c/\mathbf{x}_i, \theta^t) \left[\mathcal{H}(\mathbf{x}_i, \mathbf{w}_c) \ln(\varepsilon_c) n - \mathcal{H}(\mathbf{x}_i, \mathbf{w}_c) \ln(1 - \varepsilon_c) \right] \\ &= \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} p(c/\mathbf{x}_i, \theta^t) \left[-\ln \left(\frac{1 - \varepsilon_c}{\varepsilon_c} \right) \mathcal{H}(\mathbf{x}_i, \mathbf{w}_c) + n \ln(1 - \varepsilon_c) \right] \end{aligned}$$

Finalement on obtient :

$$\begin{aligned} Q_1^T(\theta^C,\theta^t) &= \sum_{\mathbf{x}_i\in\mathcal{A}}\sum_{c\in C}\left[-\ln\left(\frac{1-\varepsilon_c}{\varepsilon_c}\right)p(c/\mathbf{x}_i,\theta^t)\mathcal{H}(\mathbf{x}_i,\mathbf{w}_c)+np(c/\mathbf{x}_i,\theta^t)\ln(1-\varepsilon_c)\right] \\ &= \sum_{c\in C}\sum_{\mathbf{x}_i\in\mathcal{A}}\left[-\ln(\frac{1-\varepsilon_c}{\varepsilon_c})p(c/\mathbf{x}_i,\theta^t)\mathcal{H}(\mathbf{x}_i,\mathbf{w}_c)\right] \\ &\quad +\sum_{c\in C}\sum_{\mathbf{x}_i\in\mathcal{A}}\left[np(c/\mathbf{x}_i,\theta^t)\ln(1-\varepsilon_c)\right] \end{aligned}$$

Puisque $\theta_c=(\varepsilon_c,\mathbf{w}_c)$ la maximisation de la fonction $Q_1^T(\theta^C,\theta^t)$ s'effectue en deux étapes : la première consiste à fixer ε_c et maximiser par rapport à $\mathbf{w}_c$; à l'inverse dans la deuxième étape, on fixe $\mathbf{w}_c$ et on maximise par rapport à ε_c.

- **Etape 1 :** Pour une valeur fixe de ε_c de $]0,1/2[$, le deuxième terme de la fonction $Q_1^T(\theta^C,\theta^t)$ est constant et positif, donc la maximisation de $Q_1^T(\theta^C,\theta^t)$ est équivalente à la minimisation de l'inertie $\mathcal{I}_c(\mathbf{w}_c)$ au niveau de chaque cellule c :

$$\mathcal{I}_c(\mathbf{w}_c)=\sum_{\mathbf{x}_i\in\mathcal{A}}\ln\left(\frac{1-\varepsilon_c}{\varepsilon_c}\right)p(c/\mathbf{x}_i,\theta^t)\mathcal{H}(\mathbf{x}_i,\mathbf{w}_c) \tag{2.16}$$

  Le point qui minimise cette inertie est le centre médian de l'ensemble des observations $\mathbf{x}_i$ appartenant au sous-ensemble $\mathcal{A}$ (voir Annexe A). Chaque composante du vecteur $\mathbf{w}_c=(w_c^1,...,w_c^k,...,w_c^n)$ est calculée comme :

$$w_c^k=\begin{cases} 0 & \text{si } \left[\sum_{\mathbf{x}_i\in\mathcal{A}}p(c/\mathbf{x}_i,\theta^t)(1-x_i^k)\right]\geq \\ & \quad\left[\sum_{\mathbf{x}_i\in\mathcal{A}}p(c/\mathbf{x}_i,\theta^t)x_i^k\right] \\ 1 & \text{sinon} \end{cases},$$

- **Etape 2 :** Dans le cas où l'on suppose que les centres médians $\mathbf{w}_c$ sont connus et sont fixés, la maximisation du critère $Q_1^T(\theta^C,\theta^t)$ par rapport à ε_c revient à résoudre l'équation $\frac{\partial Q_1^T(\theta^C,\theta^t)}{\partial\varepsilon_c}=0$. Ce qui donne les formules suivantes :

$$\frac{\partial Q_1^T(\theta^C,\theta^t)}{\partial\varepsilon_c}=\sum_{\mathbf{x}_i\in\mathcal{A}}\left[\frac{p(c/\mathbf{x}_i,\theta^t)\mathcal{H}(\mathbf{x}_i,\mathbf{w}_c)}{\varepsilon_c(1-\varepsilon_c)}-\frac{n\varepsilon_c p(c/\mathbf{x}_i,\theta^t)}{\varepsilon_c(1-\varepsilon_c)}\right]$$

  Pour $\frac{\partial Q_1^T(\theta^C,\theta^t)}{\partial\varepsilon_c}=0$, on obtient :

$$\varepsilon_c=\frac{\sum_{\mathbf{x}_i\in\mathcal{A}}p(c/\mathbf{x}_i,\theta^t)\mathcal{H}(\mathbf{x}_i,\mathbf{w}_c)}{\sum_{\mathbf{x}_i\in\mathcal{A}}np(c/\mathbf{x}_i,\theta^t)} \tag{2.17}$$

  n est la dimension du vecteur binaire $\mathbf{x}_i$.

2.4.3 Cas général : BeSOM-ϵ

On se place dans les mêmes conditions que dans la modélisation précédente en remplaçant la valeur de ε_c par un vecteur $\epsilon_c = (\varepsilon_c^1, ..., \varepsilon_c^k, ..., \varepsilon_c^n)$ composé de n valeurs ε_c^k dépendant de chaque variable binaire x^k. Ainsi, la loi de probabilité est réécrite :

$$p(\mathbf{x}/\epsilon_c, \mathbf{w}_c) = \prod_{k=1}^{n} (\varepsilon_c^k)^{|x^k - w_c^k|} (1 - \varepsilon_c^k)^{1-|x^k - w_c^k|}$$

Pour la maximisation de $Q_1^T(\theta^C, \theta^t)$, on suppose que le paramètre θ est constitué du couple de paramètre de la loi de Bernoulli associée à la cellule c $(\mathbf{w}_c, \epsilon_c)$ composée du vecteur référent $\mathbf{w}_c$ et du vecteur de probabilité ϵ_c. Par conséquent, la fonction de coût est réécrite de la manière suivante :

$$\begin{aligned} Q_1^T(\theta^C, \theta^t) &= \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} \sum_{c^* \in C} p(c, c^*/\mathbf{x}_i, \theta^t) \ln \left(\prod_{k=1}^{n} p(x_i^k/c) \right) \\ &= \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} \sum_{k=1}^{n} p(c/\mathbf{x}_i, \theta^t) \ln(p(x_i^k/c)) \end{aligned}$$

Ainsi,

$$\begin{aligned} Q_1^T(\theta^C, \theta^t) &= \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} \sum_{k=1}^{n} \left[p(c/\mathbf{x}_i, \theta^t) \ln \left((\varepsilon_c^k)^{|x^k - w_c^k|} (1 - \varepsilon_c^k)^{1-|x^k - w_c^k|} \right) \right] \\ &= \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in C} \sum_{k=1}^{n} p(c/\mathbf{x}_i, \theta^t) \left[-\ln \left(\frac{1 - \varepsilon_c^k}{\varepsilon_c^k} \right) |x_i^k - w_c^k| + \ln(1 - \varepsilon_c^k) \right] \\ &= \sum_{k=1}^{n} \sum_{c \in C} \sum_{\mathbf{x}_i \in \mathcal{A}} \left[-p(c/\mathbf{x}_i, \theta^t) \ln \left(\frac{1 - \varepsilon_c^k}{\varepsilon_c^k} \right) |x_i^k - w_c^k| \right] \\ &\quad + \sum_{k=1}^{n} \sum_{c \in C} \sum_{\mathbf{x}_i \in \mathcal{A}} \left[p(c/\mathbf{x}_i, \theta^t) \ln(1 - \varepsilon_c^k) \right] \end{aligned}$$

Comme dans la modélisation précédente, la maximisation de la fonction $Q_1^T(\theta^C, \theta^t)$ s'effectue en deux étapes : la première consiste à fixer ε_c^k et maximiser par rapport à w_c^k ; à l'inverse dans la deuxième étape, on fixe w_c^k et on maximise par rapport à ε_c^k.

– **Etape 1 :** Pour une valeur fixe de ε_c^k de $]0, 1/2[$, le deuxième terme de la fonction $Q_1^T(\theta^C, \theta^t)$ est constant et positif, donc la maximisation de $Q_1^T(\theta^C, \theta^t)$ est équivalente à la minimisation de l'inertie $\mathcal{I}_c^k(w_c^k)$ au niveau de chaque cellule c et par rapport à chaque variable k :

$$\mathcal{I}_c^k(w_c^k) = \sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i, \theta^t) \ln\left(\frac{1-\varepsilon_c^k}{\varepsilon_c^k}\right) |x_i^k - w_c^k| \tag{2.18}$$

Le point qui minimise cette inertie est la valeur médiane de l'ensemble des observations de la variable $\mathbf{x}_i^k$ appartenant au sous-ensemble $\mathcal{A}$ (voir Annexe A). Chaque composante de $\mathbf{w}_c = (w_c^1, ..., w_c^k, ..., w_c^n)$ est calculée de la manière suivante :

$$w_c^k = \begin{cases} 0 & \text{si } \left[\sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i, \theta^{t-1})(1 - x_i^k)\right] \geq \\ & \left[\sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i, \theta^{t-1}) x_i^k\right] \\ 1 & \text{sinon} \end{cases} \tag{2.19}$$

– **Etape 2 :** Dans le cas où l'on suppose que les centres médians $\mathbf{w}_c$ sont connus et sont fixés, la maximisation du critère $Q_1^T(\theta^C, \theta^t)$ par rapport à ε_c^k revient à résoudre l'équation $\frac{\partial Q_1^T(\theta^C, \theta^t)}{\partial \varepsilon_c^k} = 0$, ce qui donne les formules suivantes :

$$\frac{\partial Q_1^T(\theta^C, \theta^t)}{\partial \varepsilon_c^k} = \sum_{\mathbf{x}_i \in \mathcal{A}} \left[\frac{p(c/\mathbf{x}_i, \theta^t)|x_i^k - w_c^k|}{\varepsilon_c^k(1-\varepsilon_c^k)} - \frac{\varepsilon_c^k p(c/\mathbf{x}_i, \theta^t)}{\varepsilon_c^k(1-\varepsilon_c^k)}\right]$$

La résolution de l'équation $\frac{\partial Q_1^T(\theta^C, \theta^t)}{\partial \varepsilon_c^k} = 0$, permet d'obtenir la formule suivante :

$$\varepsilon_c^k = \frac{\sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i, \theta^t)|x_i^k - w_{c1}^k|}{\sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i, \theta^t)} \tag{2.20}$$

où

$$p(c^*/\mathbf{x}_i, \theta^{t-1}) = \sum_{c \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1})$$

et

$$p(c/\mathbf{x}_i, \theta^{t-1}) = \sum_{c^* \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1})$$

2.4.4 Algorithme d'apprentissage BeSOM

L'algorithme d'apprentissage de BeSOM est l'application de l'algorithme EM aux modèles des cartes topologiques probabilistes adaptées aux données catégorielles codées en binaire. Cet algorithme permet d'estimer les coefficients du modèle de mélange. En supposant que les probabilités $p(c^*/c)$ sont fixées pour une valeur donnée de T, on aura

alors l'algorithme d'apprentissage BeSOM à T fixé présenté ci-dessous. Cet algorithme consiste à réitérer les deux étapes du formalisme EM présenté au paragraphe §2.3. Si on note θ^t le vecteur de paramètres, estimé lors de la $t^{ème}$ itération et θ^{t+1} la mise à jour de ce vecteur, l'algorithme BeSOM pour une température T fixe se présente de la manière suivante :

Initialisation
Partant d'une position θ^0, et un nombre d'itérations N_{iter}, l'itération de base de l'algorithme BeSOM, est définie ainsi :

Itération de base$(t \geq 1)$
Ayant estimer les paramètres θ^t à l'itération précédente, l'itération en cours estime les nouveaux paramètres θ^{t+1} et ceci en appliquant la formule (2.14) pour l'estimation du paramètre $\theta^{c^*,t+1}$ et puis déterminer l'ensemble des nouveaux référents $\mathcal{W}^{t+1}$ correspondant aux centres médians et les probabilités des lois de Bernoulli ϵ^{t+1} en utilisant l'expression (2.20) si on se trouve dans le cas général où ϵ^{t+1} est un vecteur.

Répéter l'itération de base jusqu'à $t \geq N_{iter}$

On constate que l'algorithme d'apprentissage BeSOM dépend de l'initialisation des paramètres à estimer. Les résultats obtenus dépendent assez fortement de ces conditions. Dans tous les exemples qui vont suivre, l'ensemble des paramètres $\theta_0^{\mathcal{C}^*}$ sont pris égaux à $\frac{1}{K}$ et les paramètres $\theta_0^{\mathcal{C}}$ sont initialisés à l'aide de la partition trouvée par l'algorithme des cartes topologiques déterministes dédiées aux données binaires, BinBatch [104]. Si on fait varier le paramètre T, l'algorithme d'apprentissage de BeSOM se présente de la manière suivante :

1) **Phase d'initialisation** ($t = 0$)

   – Choisir T_{max}, T_{min} et N_{iter}. Effectuer l'algorithme d'apprentissage BeSOM pour la valeur de T constante égale à T_{max}.

2) **Etape itérative**
   – L'ensemble des paramètres θ^t de l'étape précédente est connu. Calculer la nouvelle valeur de T en appliquant la formule suivante :

$$T = T_{max} \left(\frac{T_{min}}{T_{max}} \right)^{\frac{t}{N_{iter}-1}}$$

   – Pour cette valeur du paramètre T, calculer l'itération de base définie dans l'algorithme indiqué précédemment pour un paramètre fixe T.

3) **Répéter** l'étape itérative tant que $t \leq N_{iter}$

2.4.5 Lien entre l'algorithme Cartes auto-organisatrices binaires BinBatch et BeSOM

On considère ici l'algorithme batch proposé par Kohonen. Cet algorithme a été développé pour des données continues, mais un modèle spécifique a été proposé pour des données binaires utilisant une fonction de coût similaire (BinBatch) [104]. Utilisant la distance de Hamming $\mathcal{H}$ (Annexe A), la fonction de coût dans ce cas est exprimée de la manière suivante :

$$\mathcal{G}(\mathbf{w}, \phi) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{r \in \mathcal{C}} K^T(\delta(\phi(\mathbf{x}_i), r)) \mathcal{H}(\mathbf{x}_i, \mathbf{w}_r) \tag{2.21}$$

où ϕ affecte chaque observation $\mathbf{x}$ à une cellule de $\mathcal{C}$.

La fonction de coût (2.21) est minimisée en utilisant un processus itératif à deux étapes. La phase d'affectation qui utilise la fonction d'affectation suivante : $\forall \mathbf{x},\ \phi(\mathbf{x}) = \arg\min_c \mathcal{H}(\mathbf{x}, \mathbf{w}_c)$. La phase d'optimisation qui nous permet de définir le centre médian $\mathbf{w}_c \in \beta^n$ des observations $\mathbf{x}_i \in \mathcal{A}$ pondérées par $\mathcal{K}^T(\delta(c, \phi(\mathbf{x}_i)))$. Chaque composante

de $\mathbf{w}_c = (w_c^1, ..., w_c^k, ..., w_c^n)$ est ensuite calculée comme suit :

$$w_c^k = \begin{cases} 0 & \text{si } \left[\sum_{\mathbf{x}_i \in \mathcal{A}} \mathcal{K}^T(\delta(c, \phi(\mathbf{x}_i)))(1 - x_i^k)\right] \geq \\ & \left[\sum_{\mathbf{x}_i \in \mathcal{A}} \mathcal{K}^T(\delta(c, \phi(\mathbf{x}_i)))x_i^k\right] \\ 1 & \text{sinon} \end{cases} \tag{2.22}$$

La mise à jour du centre médian $\mathbf{w}_c$ dans le modèle BeSOM coïncide avec le modèle BinBatch dans lequel chaque donnée $\mathbf{x}$ est pondérée par la fonction de voisinage centrée sur le prototype gagnant $\mathbf{w}_c$ (eq. 2.22). Dans les deux modèles, l'inertie est minimisée dans un espace binaire utilisant la distance de Hamming pondérée. Cette minimisation est effectuée en calculant le centre médian binaire $\mathbf{w}_c$ qui utilise le même codage binaire que l'espace binaire des données.

On constate aussi que la définition du gagnant dans le modèle BinBatch est différente de celle du BeSOM. Dans les deux variantes BeSOM$-\varepsilon$ et BeSOM$-\epsilon$, la phase d'affectation est réalisée à la fin de l'algorithme d'apprentissage. Il existe un lien fort entre le modèle BinBatch et les modèles de mélanges. A chaque étape de l'algorithme EM, l'algorithme BeSOM calcule la probabilité a posteriori $p(c/\mathbf{x})$ qui est utilisée pour pondérer l'observation $\mathbf{x}$. Dans l'algorithme BinBatch, l'affectation est simplifiée en choisissant le plus proche prototype au sens de la distance de Hamming $\mathcal{H}(\mathbf{x}, \mathbf{w}_c)$. Afin de découvrir le lien entre le modèle probabiliste et déterministe, nous supposons que le modèle probabiliste correspond au modèle BeSOM$-\varepsilon$ où le paramètre de probabilité ϵ est le même pour toutes les variables et pour toutes les cellules $\epsilon = \varepsilon$. Nous supposons aussi que les probabilités a priori $p(c^*) = \frac{1}{K}$ sont égales, ainsi une faible distance de Hamming est similaire au cas que la probabilité de trouver $\mathbf{x}$ sachant les paramètres $(\mathbf{w}_c, \varepsilon)$ est assez élevée.

$$p(\mathbf{x}/c) = \varepsilon^{\mathcal{H}(\mathbf{x}, \mathbf{w}_c)}(1 - \varepsilon)^{n - \mathcal{H}(\mathbf{x}, \mathbf{w}_c)}$$

Par conséquent, on déduit dans ce cas que les probabilités a postériori $p(c/\mathbf{z})$ et $p(c^*/\mathbf{z})$ deviennent grandes. Dans les conditions définies ci-dessus, la maximisation de la fonction objective $Q^T(\theta^t, \theta^{t-1})$ est équivalente à la maximisation de la fonction de coût $Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1})$ par rapport à $\theta_c = (\mathbf{w}_c, \varepsilon)$, qui est réecrite de la manière suivante :

$$\begin{aligned} Q_1^T(\theta^C, \theta^{t-1}) = \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) \sum_{c \in C} \sum_{\mathbf{x}_i \in \mathcal{A}} \left[-p(c/\mathbf{x}_i, \theta^{t-1})\mathcal{H}(\mathbf{x}_i, \mathbf{w}_c)\right] \\ + \ln(1-\varepsilon) \sum_{c \in C} \sum_{\mathbf{x}_i \in \mathcal{A}} \left[np(c/\mathbf{x}_i, \theta^{t-1})\right] \end{aligned}$$

Maximiser $Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1})$ par rapport à ε implique le calcul de la dérivé de $Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1})$ par rapport à ε et resoudre l'équation $\frac{\partial Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1})}{\partial \varepsilon} = 0$. Ainsi, on peut calculer le nouveau

paramètres ε à l'étape t de la manière suivante :

$$\varepsilon = \frac{\sum_{c\in\mathcal{C}}\sum_{\mathbf{x}_i\in\mathcal{A}} p(c/\mathbf{x}_i,\theta^{t-1})\mathcal{H}(\mathbf{x}_i,\mathbf{w}_c)}{nK}$$

Maximiser $Q_1^T(\theta^{\mathcal{C}},\theta^{t-1})$ par rapport à $\mathbf{w}_c$ nécessite de minimiser la fonction de coût $G(\mathbf{w})$ définie comme :

$$G(\mathbf{w}) = \sum_{c\in\mathcal{C}}\sum_{\mathbf{x}_i\in\mathcal{A}} p(c/\mathbf{x}_i,\theta^{t-1})\mathcal{H}(\mathbf{x}_i,\mathbf{w}_c) \tag{2.23}$$

Il est clair que l'algorithme BinBatch tend à rendre toutes les cellules d'une importance égale et maximise la fonction de coût $Q(\theta^t,\theta^{t-1})$. Si on suppose que la cellule c^* est fixée avec une fonction d'affectation, la fonction de coût $\mathcal{G}(\mathbf{w},\phi)$ (2.21) est une simplification de la fonction objective $G(\mathbf{w})$ (2.23) avec les conditions décrites antérieurement.

2.5 Expérimentation

Pour évaluer la qualité de la classification, on adopte une approche d'évaluation qui utilise des étiquettes externes. Ainsi, on utilise la pureté de la classification pour mesurer les résultats de la classification. C'est une approche fréquente dans le domaine de la classification des données. En général, les résultats de la classification sont évalués en se basant sur des connaissances externes sur la façon dont les classes doivent être structurées. Cela peut impliquer de calculer des critères comme : la séparation, la densité, la connectivité, etc. La seule manière de juger l'utilité du résultat du clustering est la validation indirecte, par laquelle les classes sont appliquées à la solution d'un problème et l'exactitude est évaluée par rapport à l'objectif des connaissances externes. Cette procédure est définie par [90] comme "validation du clustering par des variables externes", et a été utilisée dans plusieurs travaux [97, 3]. Nous pensons que cette approche est raisonnable, si on ne veut pas juger les résultats de la classification par certains indices de validité de la classe, qui ne sont qu'un choix parmi certaines propriétés préférées des classes (par exemple : compact, ou bien séparés, ou connectés). Par conséquent, pour adopter cette approche on a besoin des jeux de données étiquetées, où la connaissance externe représente l'information sur la classe fournit par les étiquettes. Ainsi, si l'algorithme BeSOM trouve des classes pertinentes dans les données, cela sera reflété par la distribution des classes. Donc on utilise un vote majoritaire pour les clusters et on les compare au comportement des méthodes dans la littérature. La technique du vote majoritaire contient les étapes suivantes. Pour chaque cluster $c \in \mathcal{C}$:

– On compte le nombre d'observations de chaque classe l (appelé N_{cl}).

- On compte le nombre total d'observations affectées à la cellule c (appelé N_c).
- On calcule la proportion d'observation de chaque classe (appelée $S_{cl} = N_{cl}/N_c$).
- On attribue au cluster l'étiquette de la classe la plus représentée ($l = \arg\max_l(S_{cl})$.

Une cellule c pour laquelle $S_{cl} = 1$ pour certaine classe étiquettée l est d'habitude appelée un cluster "pure", et une mesure de pureté peut être exprimé comme le pourcentge d'éléments de la classe affectée au cluster. Les résultats expérimentaux sont ensuite exprimés comme une fraction des observations qui interviennent dans les clusters et qui sont étiquetées avec une classe différente de celle de l'observation. Cette quantité est exprimée sous forme de pourcentage et est appelée "erreur de classification"(indiquée comme $Err\%$ dans les résultats).

Des techniques de cross-validation ou de re-échantillonage, cependant, peuvent être très utiles pour pour juger la stabilité de l'algorithme, en comparant la structure de la classification dans des expériences répétées.

2.5.1 Validation expérimentale de l'approche BeSOM

2.5.1.1 La base Zoo

C'est une base de données extraite du répertoire UCI [24]. On utilise ce simple jeu de données pour montrer les bonnes performances de notre algorithme BeSOM. Ce jeu de données contient l'information sur 101 animaux décrits par 16 variables qualitatives : dont 15 variables sont binaires et une est catégorielle avec 6 modalités. Chaque animal est étiquetté de 1 à 7 conformément à sa classe (son espèce). Utilisant le codage disjonctif (Annexe A) pour les variables qualitatives avec 6 valeurs possibles, définit dans le tableau 2.2, on obtient une matrice binaire 101×21 ($individus \times variables$).

Pour l'apprentissage du modèle général BeSOM$-\epsilon$, nous avons utilisé une carte de dimensions 5×5 cellules. L'algorithme d'apprentissage nous fournit les paramètres de la distribution de Bernoulli pour chaque cellule, $\mathbf{w}_c = (w_c^1, w_c^2, ..., w_c^k, ..., w_c^{21}) \in \beta^n$ et un vecteur de probabilité $\epsilon_c = (\varepsilon_c^1, \varepsilon_c^2, ..., \varepsilon_c^k, ..., \varepsilon_c^{21})$.

A la fin de la phase d'apprentissage, chaque observation, qui correspond à un animal, est affectée à la cellule avec la plus grande probabilité a posteriori $p(c/\mathbf{x})$. Pour visualiser la cohérence de la carte avec les étiquettes des animaux, la figure 2.3 nous montre

antelope buffalo deer elephant giraffe hare mole opossum oryx vole (1)	dolphin porpoise (1)	bass catfish chub dogfish herring pike piranha stingray tuna (4)	carp haddock seahorse sole (4)	clam seawasp (7)
aardvark bear boar cheetah leopard lion lynx mink mongoose polecat puma pussycat raccoon wolf (1)	frog frog newt toad tuatara (5)	pitviper slowworm (3)	slug worm (7)	crab crayfish lobster octopus starfish (7)
calf cavy goat hamster pony reindeer (1)	scorpion (7)	kiwi ostrich penguin rhea vulture (2)	girl seal sealion (1)	platypus seasnake tortoise (3)
fruitbat squirrel vampire (1)	duck flamingo swan (2)	chicken dove lark parakeet pheasant sparrow wren (2)	flea termite (6)	gnat (6)
gorilla wallaby (1)	crow gull hawk skimmer skua (2)	honeybee wasp (6)	housefly moth (6)	Ladybird (6)

FIG. 2.3 – 5×5 La carte BeSOM. On montre les animaux associés à la plus grande probabilité. Le numéro entre parenthèses indique l'étiquette de la classe pour chaque cellule après l'application de la règle du vote majoritaire

Modalité	Codage binaire
1	1 0 0 0 0 0
2	0 1 0 0 0 0
3	0 0 1 0 0 0
4	0 0 0 1 0 0
5	0 0 0 0 1 0
6	0 0 0 0 0 1

TAB. 2.2 – Variable catégorielle avec 6 valeurs possibles. Chaque modalité est codée de la même manière utilisant le codage disjonctif complet.

le numéro de la classe qui correspond à chaque cellule après l'application du vote majoritaire pour chaque cellule. On observe que les cellules qui ont la même classe sont proches l'une de l'autre sur la carte. On voit aussi que les insectes qui ont l'étiquette "6" sont affectés vers la partie droite en bas de la carte. On peut faire la même analyse pour le reste des "groupes".

La figure 2.4 nous montre l'évolution de la fonction objective $Q^T(\theta^t, \theta^{t-1})$ (eq. 2.10) sans calculer le terme constant $Q_3^T(\theta^{t-1})$ et variant T de $T_{max} = 2$ à $T_{min} = 0.5$. On visualise aussi les termes $Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1})$ et $Q_2^T(\theta^{\mathcal{C}^*}, \theta^{t-1})$. La maximisation de la fonction objective est évidente.

2.5.2 La base de données des chiffres manuscrits

Cette expérience concerne une base de données composées de chiffres manuscrits ("0"−"9") extraits à partir d'une collection de cartes des services hollandaises [24]. On a 200 exemples pour chaque caractère, ainsi on a au total 2000 exemples. Chaque exemple est une imagette binaire (pixel "noir" ou "blanc") de dimension 15×16. L'ensemble de données forme une matrice binaire de dimension 2000×240. Chaque variable qualitative est un pixel à deux valeurs possibles "On=1" and "Off=0". L'algorithme BeSOM$-\epsilon$ associe à chaque cellule une distribution de Bernoulli avec un vecteur référent $\mathbf{w}_c = (w_c^1, w_c^2, ..., w_c^k, ..., w_c^{240}) \in \beta^n$ et un vecteur de probabilités $\epsilon_c = (\varepsilon_c^1, \varepsilon_c^2, ..., \varepsilon_c^k, ..., \varepsilon_c^{240})$.

La figure 2.5 nous montre les paramètres estimés par notre modèle BeSOM. La fi-

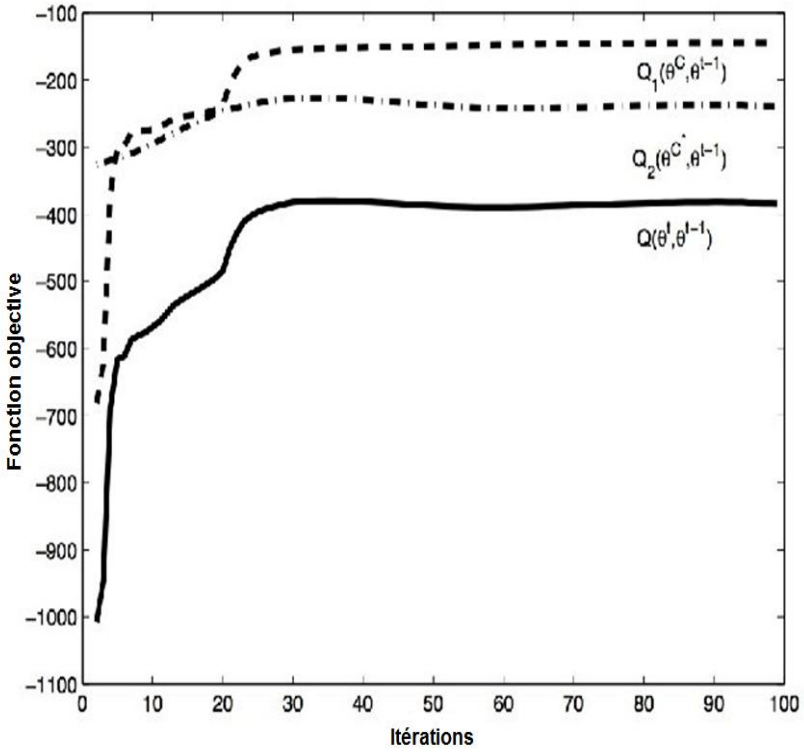


FIG. 2.4 – L'évolution de la fonction de coût $Q^T(\theta^t, \theta^{t-1})$ au cours de l'apprentissage pour le jeu de données Zoo. La ligne en pointillée représente la fonction $Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1})$. La ligne tiret-point représente la fonction $Q_2^T(\theta^{\mathcal{C}^*}, \theta^{t-1})$

gure 2.5.(a) nous montre le vecteur prototype $\mathbf{w}_c \in \beta^n$ qui est représenté comme une imagette de dimension 15×16. La figure 2.5.(b) nous montre le vecteur paramètre ϵ_c comme une imagette de la même dimension 15×16. Ainsi, chaque pixel définit la probabilité ε_c^k d'être différent de la variable binaire w_c^k associée au vecteur prototype binaire $\mathbf{w}_c$ représenté dans la figure 2.5.(a). La nuance de gris de chaque pixel est proportionnelle à la probabilité d'être différent du prototype estimé $\mathbf{w}_c$. On constate que les composantes ou les probabilités ε_c^k, qui correspondent au contour de l'image, sont élevées. En observant les deux figures à la fois, on observe une organisation topologique des prototypes assez claire sur toute la carte.

Pour étudier l'influence du choix de la loi de Bernoulli, nous avons réalisé un apprentissage de l'algorithme BeSOM-ε_c qui suppose que le paramètre ε_c dépend uniquement d'une cellule $c \in \mathcal{C}$. Au lieu d'estimer le vecteur de probabilités ϵ_c, le modèle BeSOM calcule une probabilité scalaire ε_c pour chaque cellule. La figure 2.6 nous montre les paramètres estimés par le modèle BeSOM-ε. La figure 2.6.(a) montre le vecteur proto-

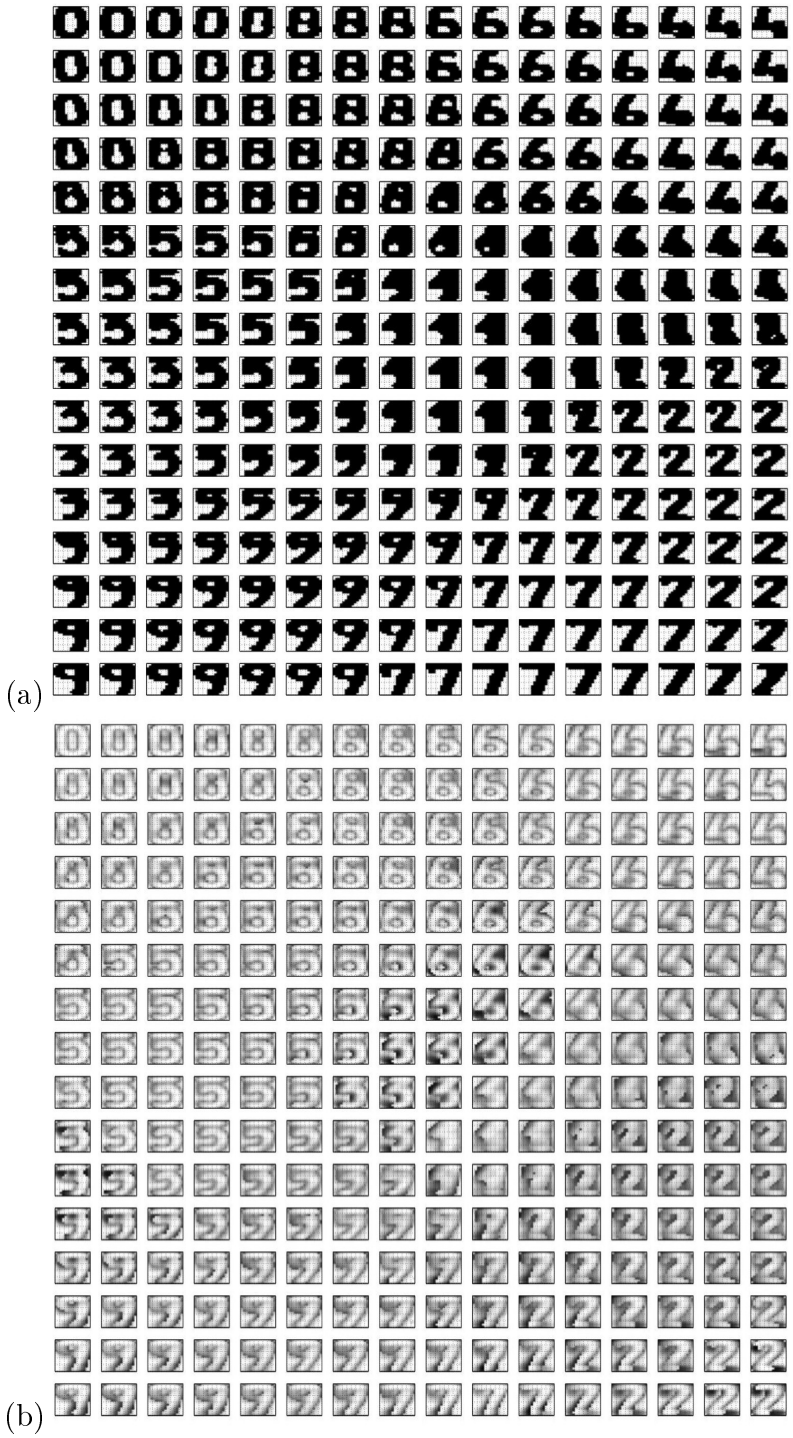


FIG. 2.5 – La carte BeSOM$-\epsilon$ de dimension 16×16. **(a)**. Chaque image est le vecteur prototype $\mathbf{w}_c = (w_c^1, w_c^2, ...w_c^k, ..., w_c^{240})$ qui est un vecteur binaire. Les images des prototypes sont bien organisées. Le pixel blanc indique "0" et le pixel noir indique "1". **(b)**.Chaque image représente un vecteur de probabilité $\epsilon_c = (\varepsilon_c^1, \varepsilon_c^2, ..., \varepsilon_c^k, ..., \varepsilon_c^{240})$.

type $\mathbf{w}_c \in \beta^n$ qui est représenté comme une imagette de dimension 15×16. On observe une organisation topologique des prototypes assez nette. La figure 2.6.(b) représente le paramètre ε_c estimé pour chaque cellule $c \in \mathcal{C}$. Ainsi, la nuance de gris de chaque cellule est proportionnelle à la probabilité ε_c d'être différente du vecteur prototype binaire $\mathbf{w}_c$ représenté dans la figure 2.6.(a).

Evidemment, on constate que le modèle général BeSOM$-\epsilon$ présenté dans la figure 2.5 fournit plus d'informations que le modèle réduit BeSOM-ε_c (figure 2.6). Le modèle BeSOM-ε_c est intéressant quand on manipule des grandes bases de données.

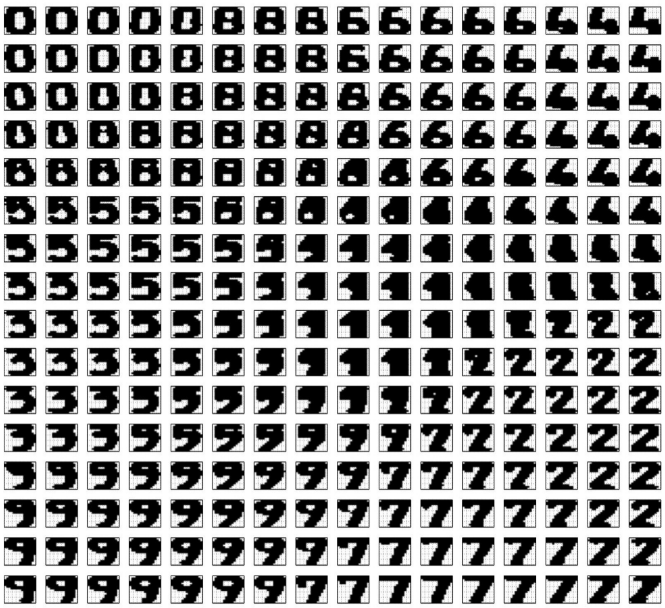
2.5.3 Différents modèles : BeSOM

Avec le modèle qu'on propose, on peut obtenir différents niveaux de pertinences. En fonction de l'hypothèse sur la probabilité ε on peut avoir trois cas :

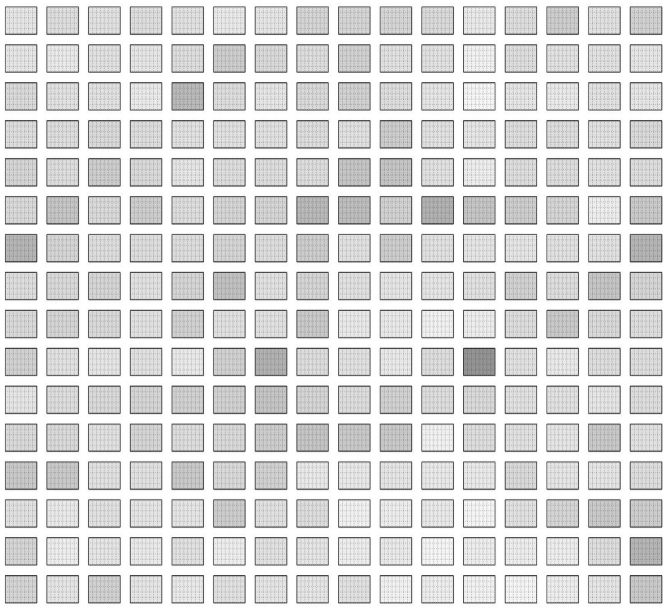
1) La probabilité ε dépend de la cellule c et de la variable j : $\epsilon_c = (\varepsilon_c^1, \varepsilon_c^2, ..., \varepsilon_c^j, ..., \varepsilon_c^n)$ figure2.7(a)**(cas général)**
2) La probabilité ϵ dépend d'une cellule seulement $\epsilon = \varepsilon_c$, figure 2.7(b)
3) La probabilité ε dépend seulement de la carte $\mathcal{C}$ (une seule valeur pour toute la carte), figure 2.7(c)

Nous observons que les valeurs des probabilités permettent de détecter les variables pertinentes. On arrive visuellement à reconnaître les chiffres. Par conséquent, il est possible d'utiliser ces valeurs pour caractériser les groupes. Dans le deuxième cas, il n'est pas possible de caractériser la forme des chiffres, mais nous pouvons caractériser les groupes. Ainsi, il est possible de sélectionner des groupes d'individus et de réaliser un échantillonnage optimisé.

Dans le troisième cas, une seule valeur réelle est estimée pour toute la carte. Afin de montrer l'intérêt de ce type de modèle nous avons appris plusieurs cartes avec des initialisations différentes et par la suite nous avons calculé la pureté associée à chacune de ces cartes. La figure 2.7(c) indique la variation de la pureté selon la probabilité ε estimée par le modèle. Plus la probabilité est faible plus le modèle est mieux. Nous observons que les modèles estimant des probabilités supérieures à 0.5 fournissent des puretés faibles. A l'opposé, ceux estimant des probabilités inférieures à 0.5 correspondent à des



(a)



(b)

FIG. 2.6 – La carte de BeSOM-ε de dimension 16×16 . Dans la figure (a) chaque image représentée est le vecteur prototype $\mathbf{w}_c = (w_c^1, w_c^2, ...w_c^k, ..., w_c^{240})$ qui est un vecteur binaire. Les prototypes des images sont bien organisés. Le pixel blanc indique "0" et le pixel noir indique "1". Dans la figure (b) chaque image représentée est la probabilité ε_c. Le niveau de gris de chaque cellule est proportionnel à la probabilité d'être différent du prototype estimé $\mathbf{w}_c$ (figure(b))

puretés fortes. Il convient donc de choisir comme meilleur modèle celui qui est associé à la valeur de probabilité la plus faible. Ce type d'application montre qu'il est possible d'utiliser ce principe pour faire de la sélection de modèles.

2.5.4 Autres bases de données

Nous avons comparé notre modèle BeSOM avec la version classique des cartes topologiques binaires. Nous avons utilisé deux autres bases :

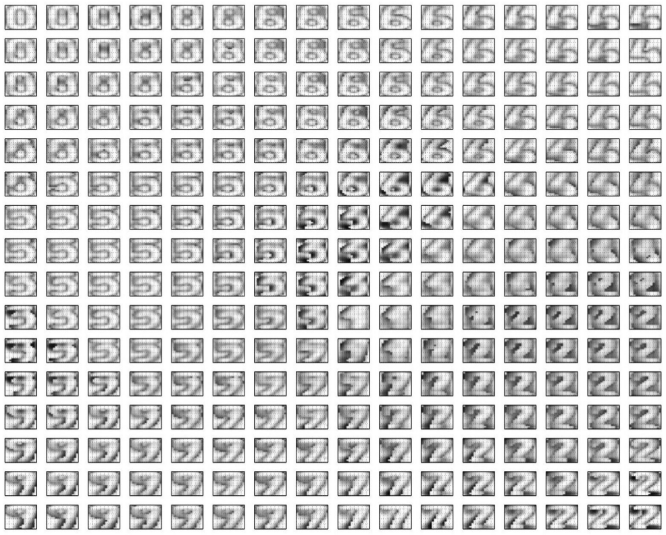
La maladie du cancer du sein (Wisconsin Breast Cancer Data)
Ce jeu de données contient 699 observations avec 9 variables. Chaque observation est étiquetée bénigne (458 ou 65.5%) ou maligne (241 ou 34.5%). Dans notre cas, toutes les variables sont considérées catégorielles avec valeurs entre $1, 2, ..., 10$ et codées en binaire (Annexe A).

La base de données "Vote" (Congressional Vote Data)
C'est un jeu de données qui contient 435 observations, dont 168 appartiennent à une classe et 267 appartiennent à la deuxième classe.

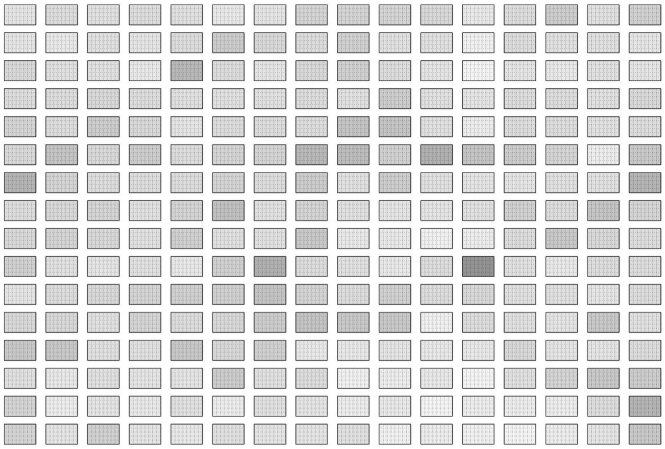
Le tableau 2.3 indique le taux de l'erreur de classification obtenu avec les deux méthodes. Nous pouvons observer que les résultats sont généralement similaires utilisant le modèle BinBatch et BeSOM$-\epsilon$, bien qu'ils sont meilleurs avec BeSOM$-\epsilon$. Comme nous l'avons signalé dans les sections précédentes le modèle BeSOM fournit plus d'informations que le modèle déterministe BinBatch (qui représente les cartes classiques utilisant la distance de Hamming).

Le jeu de données / *Err*	BinBatch	**BeSOM$-\epsilon$**
Wisconsin-B-C	3.87%	**2.43%**
Zoo	2.97%	**1.98%**
Congressional vote	5.91%	**5.77%**

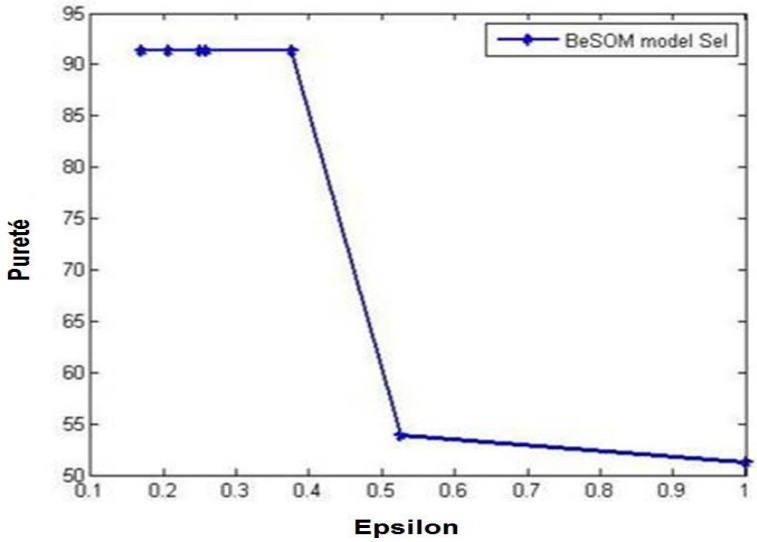
TAB. 2.3 – La comparaison des perfomances de classification de BeSOM$-\epsilon$, et BinBatch.



(a)



(b)



(c)

FIG. 2.7 – Dans la figure (a) chaque image représente un vecteur de probabilité. Dans la figure (b) on voit la probabilité par cellule, la figure (c) on voit la probabilité pour toute la carte.

2.6 Conclusion

Ce chapitre présente une nouvelle formalisation des cartes auto-organisatrices pour la classification topologique et probabiliste de données binaires multivariées ou de données catégorielles utilisant le codage binaire. Nous avons proposé dans ce chapitre un formalisme probabiliste dédié à la classification des données binaires dans lequel les cellules de la carte sont représentées par une distribution de Bernoulli. Chaque cellule est caractérisée par un prototype avec le même codage binaire de l'espace d'entrée et par la probabilité d'être différent de ce prototype. Nous avons utilisé l'algorithme EM pour maximiser la vraisemblance des données afin d'estimer les paramètres du modèle de mélange de lois de Bernoulli. Dans le chapitre suivant, nous allons présenter une extension du modèle BeSOM pour des données mixtes (quantitatives et qualitatives).

Chapitre 3

Classification probabiliste topologique de données mixtes

3.1 Introduction

On s'intéresse dans ce chapitre au traitement des variables mixtes (qualitatives et continues). On cherche en particulier des algorithmes à base des cartes auto-organisatrices permettant de traiter les particularités attachées à ces données. Dans le cas des variables mixtes, on cherche à mettre en évidence la typologie des modalités avec les variables continues et on essaye de faire ressortir les relations qui existent entre les différentes variables. Le modèle BeSOM présenté au chapitre précédent permet de proposer une classification probabiliste automatique dédiée aux données binaires. La carte topologique estime alors ses paramètres en respectant le codage binaire des données, ce qui permet une représentation binaire, interprétable de la quantification vectorielle proposée en fin d'apprentissage. Nous présentons dans ce chapitre une extension du modèle probabiliste BeSOM dédié aux données binaires. Le modèle que nous proposons est aussi une extension du modèle PrSOM qui est l'un des premiers modèles probabilistes dédiés aux données continues. Pour définir notre modèle probabiliste manipulant des données mixtes, nous avons modélisé la distribution par un mélange de mélange de lois de Bernoulli et de lois Gaussienne. Nous nous sommes inspiré du modèle *Multimix* [109, 86]. Les auteurs définissent leur modèle de "clustering" comme une généralisation commune tant des modèles "latent" que des modèles de mélange de distributions normales. Ils ont proposé une approche basée sur une forme d'indépendance locale en partitionnant les variables en plusieurs sous-vecteurs.

3.2 Modèle de mélanges des cartes topologiques

Comme le modèle des cartes topologiques défini précédemment, nous supposons que nous disposons d'une carte discrète $\mathcal{C}$ ayant K cellules structurées par un graphe non orienté. Cette structure de graphe permet de définir une distance, $\delta(r, c)$ entre deux cellules r et c de $\mathcal{C}$, comme étant la longueur de la plus courte chaine permettant de relier les cellules r et c. Le modèle que nous proposons dans ce chapitre est baptisé PrMTM (Probabilistic Topological Map) associe à chaque cellule $c \in \mathcal{C}$ une probabilité mixte $p(\mathbf{x}/c)$ où $\mathbf{x}$ est un vecteur de l'espace des données mixtes. En suivant le formalisme bayésien, introduit dans [108] et dans le chapitre 2 (section 2.2), on suppose que les observations $\mathbf{x}$ sont générées de la manière suivante : on commence par choisir une cellule c^* de $\mathcal{C}$ qui permet de déterminer un voisinage de cellule suivant la probabilité conditionnelle $p(c/c^*)$ qui est supposée connue. Ainsi l'observation $\mathbf{x}$ est générée suivant la probabilité $p(\mathbf{x}/c)$. Ce formalisme nous amène à définir le générateur des données $p(\mathbf{x})$ par un mélange de probabilités :

$$p(\mathbf{x}) = \sum_{c,c^* \in \mathcal{C}} p(c, c^*, \mathbf{x})$$

Afin de simplifier le calcul, nous supposons que le processus vérifie la propriété de "Markov", ainsi $p(\mathbf{x}/c^*, c) = p(\mathbf{x}/c)$ et $p(c^*/c, \mathbf{x}) = p(c^*/c)$.

$$p(\mathbf{x}) = \sum_{c,c^* \in \mathcal{C}} p(\mathbf{x}/c)p(c/c^*)p(c^*) = \sum_{c^* \in \mathcal{C}} p(c^*)p_{c^*}(\mathbf{x}),$$

avec

$$p_{c^*}(\mathbf{x}) = p(\mathbf{x}/c^*) = \sum_{c \in \mathcal{C}} p(c/c^*)p(\mathbf{x}/c)$$

Ainsi, $p(\mathbf{x})$ apparaît comme un mélange des probabilités $p_{c^*}(\mathbf{x})$. Les coefficients du mélange sont les probabilités $p(c^*)$.

Comme pour le modèle BeSOM, la notion de voisinage sur la carte, est définie comme suit :

$$p(c/c^*) = \frac{\mathcal{K}^T(\delta(c, c^*))}{\sum_{r \in C} \mathcal{K}^T(\delta(r, c^*))}$$

, où $\mathcal{K}^T$ est une fonction de voisinage dépendant du paramètre T et qui peut s'exprimer par $\mathcal{K}^T(\delta) = \mathcal{K}(\delta/T)$ où $\mathcal{K}$ est une fonction noyau particulière.

Nous supposons par la suite que les données $\mathbf{x}$ sont des vecteurs à d composantes composées de deux parties : une partie quantitative $\mathbf{x}_i^{r[.]} = (x_i^{r[1]}, x_i^{r[2]}, ..., x_i^{r[n]})$ $(\mathbf{x}_i^{r[.]} \in \mathcal{R}^n)$ et d'une partie qualitative codée en binaire $\mathbf{x}_i^{b[.]} = (x_i^{b[1]}, x_i^{b[2]}, ..., x_i^{b[k]}, ..., x_i^{b[m]})$ où la $k$ième composante $x_i^{b[k]}$ est une variable binaire $(x_i^{b[k]} \in \beta = \{0, 1\})$. Chaque observation $\mathbf{x}_i$ est ainsi la réalisation d'une variable aléatoire appartenant à $\mathcal{R}^n \times \beta^m$. Avec ces notations une observation particulière $\mathbf{x}_i = (\mathbf{x}_i^{r[.]}, \mathbf{x}_i^{b[.]})$ est un vecteur composé d'une partie quantitative et une binaire avec la dimension $d = n + m$. Nous rappelons que $\mathcal{A}$ représente l'ensemble des observations de taille N.

Afin de simplifier ce modèle, nous supposons que la composante quantitative est indépendante de la composante binaire. Ainsi la probabilité conditionnelle peut être réécrite comme le produit de deux termes :

$$p(\mathbf{x}/c) = p(\mathbf{x}^{r[.]}/c) \times p(\mathbf{x}^{b[.]}/c)$$

Nous prenons aussi une hypothèse forte qui est de considérer que les n composantes quantitatives et les m binaires sont indépendantes sachant une cellule c :

$$p(\mathbf{x}^{b[.]}/c) = \prod_{k=1}^{m} p(x^{b[k]}/c)$$

$$p(\mathbf{x}^{r[.]}/c) = \prod_{k=1}^{n} p(x^{r[k]}/c)$$

Nous associons dans la suite à chaque cellule $c \in \mathcal{C}$ de la carte une probabilité conditionnelle qui décrit la génération d'observation par une cellule c avec les deux parties :

- La fonction densité de la partie quantitative qui est une gaussienne sphérique $p(\mathbf{x}^{r[.]}/c) = \mathcal{N}(\mathbf{x}^{r[.]}, \mathbf{w}^{r[.]}, \sigma_c^2 I)$, définie par "la moyenne" (vecteur référent $\mathbf{w}^{r[.]} = (w^{r[1]}, ..., w^{r[k]}, w^{r[n]})$), la matrice covariance, définie par $\sigma_c^2 I$ où σ_c est l'écart type et I la matrice identité,

$$\mathcal{N}(\mathbf{x}^{r[.]}, \mathbf{w}^{r[.]}, \sigma_c^2 I) = \frac{1}{(2\pi\sigma_c)^{\frac{n}{2}}} \exp\left[\frac{||\mathbf{w}_c^{r[.]} - \mathbf{x}_i^{r[.]}||^2}{2\sigma_c^2}\right] \tag{3.1}$$

- La fonction densité de la partie binaire qui est la distribution de Bernoulli $p(\mathbf{x}^{b[.]}/c) = p(\mathbf{x}^{b[.]}/\varepsilon_c, \mathbf{w}_c^{b[.]}) = f_c(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]}, \varepsilon_c)$ avec les paramètres $\mathbf{w}_c^{b[.]} = (w_c^{b[1]}, w_c^{b[2]}, ..., w_c^{b[k]}, ..., w_c^{b[m]}) \in$

β^m avec la probabilité $\varepsilon_c \in]0, \frac{1}{2}[$ associée à $\mathbf{w}_c^{b[.]} \in \{0,1\}^m$ (chapitre 2, §2.4). Le paramètre ε_c définit la probabilité d'être différent du prototype $\mathbf{w}_c^{b[.]}$. La distribution associée à chaque cellule $c \in \mathcal{C}$ est celle utilisée par le modèle BeSOM qui est définie comme suit :

$$f_c(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]}, \varepsilon_c) = \varepsilon_c^{\mathcal{H}(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]})}(1-\varepsilon_c)^{m-\mathcal{H}(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]})} \tag{3.2}$$

$\mathcal{H}$ est la distance de Hamming qui permet la comparaison de deux vecteurs binaires. Cette distance mesure le nombre de composantes binaires différentes entre $\mathbf{x}_i^{b[.]}$ et $\mathbf{x}_j^{b[.]}$:

$$\mathcal{H}(\mathbf{x}_i^{b[.]}, \mathbf{x}_j^{b[.]}) = \sum_{k=1}^{m} |x_i^{b[k]} - x_j^{b[k]}|$$

Les paramètres $\theta = \theta^{\mathcal{C}} \cup \theta^{\mathcal{C}^*}$ qui définissent le modèle générateur de mélange sont constitués à la fois des paramètres de la distribution Gaussienne et Bernoulli ($\theta^{\mathcal{C}} = \{\theta^c, c = 1..K\}$, où $\theta^c = (\mathbf{w}_c = (\mathbf{w}_c^{r[.]}, \mathbf{w}_c^{b[.]}), \sigma_c^2, \varepsilon_c)$), et la probabilité a priori aussi appellée coefficient de la mixture ($\theta^{\mathcal{C}^*} = \{\theta_{c^*}, c^* = 1..K\}$, où $\theta_{c^*} = p(c^*)$). L'objectif maintenant est de définir la fonction de coût ainsi que l'algorithme d'apprentissage associé qui permettra d'estimer ces paramètres.

3.2.1 Algorithme d'apprentissage

De la même manière que l'algorithme BeSOM, l'algorithme d'apprentissage consiste à maximiser la vraisemblance des observations en appliquant l'algorithme EM. L'usage de l'algorithme EM s'explique par l'existence d'une variable cachée notée $\mathbf{z}$, constituée par le couple de cellule c et c^*, $\mathbf{z} = (c, c^*)$, impliquées dans la génération d'une donnée observée $\mathbf{x}$. En effet, la variable cachée $\mathbf{z} = (c, c^*)$ apparaît lorsqu'on écrit :

$$\begin{aligned} p(\mathbf{x}) &= \sum_{\mathbf{z} \in \mathcal{C} \times \mathcal{C}} p(\mathbf{x}, \mathbf{z}) \\ &= \sum_{c, c^* \in \mathcal{C}} p(\mathbf{x}/c) p(c^*) p(c/c^*) \mathcal{N}(\mathbf{x}^{r[.]}, \mathbf{w}^{r[.]}, \sigma_c^2 I) \times f_c(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]}, \varepsilon_c) \end{aligned} \tag{3.3}$$

A chaque donnée mixte réellement observée $\mathbf{x}$, correspond une variable indicatrice non observée $\mathbf{z}$ qui appartient à $\mathcal{C} \times \mathcal{C}$ qui permet d'identifier le couple de cellules responsables de la génération de l'observation $\mathbf{x}_i$. On note par $\mathbf{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_N\}$ l'ensemble des variables non observées.

$$z_i^{(c,c^*)} = \begin{cases} 1 & \text{pour } \mathbf{z}_i = (c, c^*) \\ 0 & \text{sinon} \end{cases} \tag{3.4}$$

Donc, nous pouvons définir la vraisemblance des données de la façon suivante :

$$V^T(\mathcal{A}, \mathbf{Z}; \theta) = \prod_{i=1}^{K} \prod_{c^* \in \mathcal{C}} \prod_{c \in \mathcal{C}} \left[p(c^*) p(c/c^*) \mathcal{N}(\mathbf{x}_i^{r[.]}, \mathbf{w}^{r[.]}, \sigma_{c_1}^2 I) \times f_c(\mathbf{x}_i^{b[.]}, \mathbf{w}_c^{b[.]}, \varepsilon_c) \right]^{z_i^{(c,c^*)}} \quad (3.5)$$

L'application de l'algorithme EM pour la maximisation de la vraisemblance des données observées nécessite d'une part l'estimation de

$$Q^T(\theta, \theta^t) = E\left[\ln V^T(\mathcal{A}, \mathbf{Z}; \theta) / \mathcal{A}, \theta^t \right],$$

où θ^t est l'ensemble des paramètres estimés à la $t^{ème}$ itération de l'algorithme, et θ l'ensemble des paramètres recherchés.

L'étape "E (Estimation)" calcule l'espérance du log-vraisemblance par rapport aux variables cachées en prenant en considération les paramétres θ^{t-1}. A l'issu de l'étape "M (Maximisation)", la fonction $Q^T(\theta^t, \theta^{t-1})$ est maximisée par rapport θ^t, ($\theta^t = \arg\max_\theta(Q^T(\theta, \theta^{t-1}))$). Ces deux étapes maximisent la fonction objective $Q^T(\theta^t, \theta^{t-1})$ où $\ln V^T(\mathcal{A}, \theta^t) \geq \ln V^T(\mathcal{A}, \theta^{t-1})$. Ainsi, la fonction $Q^T(\theta^t, \theta^{t-1})$ est définie de la manière suivante :

$$Q^T(\theta^t, \theta^{t-1}) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c^* \in \mathcal{C}} \sum_{c \in \mathcal{C}} E(z_i^{(c,c^*)} / \mathbf{x}_i, \theta^{t-1})$$
$$\left(\ln(\theta^{c^*}) + \ln\left(\frac{\mathcal{K}^T(\delta(c^*, c))}{T_{c^*}} \right) + \ln\left(\mathcal{N}(\mathbf{x}^{r[.]}, \mathbf{w}_c^{r[.]}, \sigma_c^2 I) \times f_c(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]}, \varepsilon_c) \right) \right) \quad (3.6)$$

où $T_{c^*} = \sum_{r \in \mathcal{C}} \mathcal{K}^T(\delta(r, c^*))$ et la variable $z_i^{(c,c^*)}$ est une variable de Bernoulli.

En suivant le même développement comme dans le chapitre précédent, l'espérance est définie comme suit :

$$E(z_i^{(c,c^*)} / \mathbf{x}_i, \theta^{t-1}) = p(z_i^{(c,c^*)} = 1 / \mathbf{x}_i, \theta^{t-1}) = p(c, c^* / \mathbf{x}_i, \theta^{t-1})$$

où

$$p(c, c^* / \mathbf{x}_i, \theta^{t-1}) = \frac{p(c^*) p(c/c^*) p(\mathbf{x}/c)}{p(\mathbf{x})}$$

Ainsi la fonction $Q^T(\theta^t, \theta^{t-1})$ s'écrit :

$$\begin{aligned} Q^T(\theta^t, \theta^{t-1}) &= \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in \mathcal{C}} \sum_{c^* \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1}) \ln(\mathcal{N}(\mathbf{x}^{r[.]}, \mathbf{w}_c^{r[.]}, \sigma_c^2 I)) \\ &+ \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in \mathcal{C}} \sum_{c^* \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1}) \ln(f_c(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]}, \varepsilon_c)) \\ &+ \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c^* \in \mathcal{C}} \sum_{c \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1}) \ln(\theta^{c^*}) \\ &+ \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c^* \in \mathcal{C}} \sum_{c \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1}) \ln\left(\frac{\mathcal{K}^T(\delta(c^*, c))}{T_{c^*}}\right) \end{aligned}$$

Les variables θ^c et θ^{c^*} indiquent les paramètres estimés à l'itération t. Nous observons que la fonction objective (3.7) $Q^T(\theta^t, \theta^{t-1})$ est définie comme une somme de trois termes principaux. Le premier terme $Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1})$ est composé de deux parties qui dépendent respectivement des paramètres $(\mathbf{w}_c^{r[.]}, \sigma_c)$ et $(\mathbf{w}_c^{b[.]}, \varepsilon_c)$; le second terme $Q_2^T(\theta^{\mathcal{C}^*}, \theta^{t-1})$ dépend du paramètre θ^{c^*}, et le troisième terme est constant. La maximisation de la fonction $Q^T(\theta^t, \theta^{t-1})$ par rapport à θ^{c^*} et θ^c est réalisée séparément. Ainsi,

$$Q^T(\theta^t, \theta^{t-1}) = Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1}) + Q_2^T(\theta^{\mathcal{C}^*}, \theta^{t-1}) + Q_3^T(\theta^{t-1}) \tag{3.7}$$

où

$$Q_2^T(\theta^{\mathcal{C}^*}, \theta^{t-1}) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c^* \in \mathcal{C}} \sum_{c \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1}) \ln(\theta^{c^*})$$

et

$$Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1}) = Q_1^{r[.],T}(\mathbf{w}^{r[.]}, \sigma_c, \theta^{t-1}) + Q_1^{b[.],T}(\mathbf{w}^{b[.]}, \varepsilon_c, \theta^{t-1})$$

$$Q_1^{r[.],T}(\mathbf{w}^{r[.]}, \sigma_c, \theta^{t-1}) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in \mathcal{C}} \sum_{c^* \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1}) \ln(\mathcal{N}(\mathbf{x}^{r[.]}, \mathbf{w}_c^{r[.]}, \sigma_c^2 I))$$

$$Q_1^{b[.],T}(\mathbf{w}^{b[.]}, \varepsilon_c, \theta^{t-1}) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in \mathcal{C}} \sum_{c^* \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1}) \ln(f_c(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]}, \varepsilon_c))$$

3.2.1.1 Maximisation de la fonction $Q^T(\theta^t, \theta^{t-1})$

Maximiser la fonction (3.7) revient à maximiser les trois premiers termes séparément. Le terme $Q_1^{r[.],T}(\mathbf{w}^{r[.]}, \sigma_c, \theta^{t-1})$ par rapport à $\mathbf{w}^{r[.]}$ et σ_c, la fonction $Q_1^{b[.],T}(\mathbf{w}^{b[.]}, \varepsilon_c, \theta^{t-1})$ par rapport à ε_c et θ^{t-1}. Finalement le terme $Q_2^T(\theta^{\mathcal{C}^*}, \theta^{t-1})$ par rapport à $\theta^{\mathcal{C}^*}$.

- **Maximiser** $Q_2^T(\theta^{\mathcal{C}^*}, \theta^{t-1})$:

$$Q_2^T(\theta^{\mathcal{C}^*}, \theta^{t-1}) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c^* \in \mathcal{C}} \sum_{c \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1}) \ln(\theta^{c^*})$$

  Afin d'estimer les paramètres $\theta^{\mathcal{C}^*}$ qui est l'ensemble des probabilités a priori, nous procédons de la même manière que dans le chapitre 2, section (3.2.1) qui consiste à donner une forme explicite aux probabilités a priori. Ainsi la probabilité a priori est estimée avec l'équation suivante :

$$\theta^{c^*} = p(c^*) = \frac{\sum_{\mathbf{x}_i \in \mathcal{A}} p(c^*/\mathbf{x}_i, \theta^{t-1})}{K} \tag{3.8}$$

- **Maximiser** $Q_1^{r[.],T}(\mathbf{w}^{r[.]}, \sigma_c, \theta^{t-1})$:

$$Q_1^{r[.],T}(\mathbf{w}^{r[.]}, \sigma_c, \theta^{t-1}) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in \mathcal{C}} \sum_{c^* \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1}) \ln(\mathcal{N}(\mathbf{x}^{r[.]}, \mathbf{w}_c^{r[.]}, \sigma_c^2 I))$$

  La maximisation de cette fonction nécessite une maximisation en deux étapes. La première consiste à fixer le paramètre $\mathbf{w}_c^{r[.]}$ et à maximiser la fonction $Q_1^{r[.],T}(\mathbf{w}^{r[.]}, \varepsilon_c, \theta^{t-1})$ en dérivant la fonction par rapport à σ_c. Et la deuxième consiste à fixer l'écart σ_c et à dériver par rapport à $\mathbf{w}_c^{r[.]}$:

$$\mathbf{w}_c^{r[.]} = \frac{\sum_{\mathbf{x}_i \in \mathcal{A}} \mathbf{x}_i^{r[.]} p(c/\mathbf{x}_i)}{\sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i)} \tag{3.9}$$

$$\sigma_c^2 = \frac{\sum_{\mathbf{x}_i \in \mathcal{A}} ||\mathbf{w}_c^{r[.]} - \mathbf{x}_i^{r[.]}||^2 p(c/\mathbf{x}_i)}{n \sum_{\mathbf{x} \in \mathcal{A}} p(c/\mathbf{x}_i)} \tag{3.10}$$

- **Maximiser** $Q_1^{b[.],T}(\mathbf{w}^{b[.]}, \varepsilon_c, \theta^{t-1})$

$$Q_1^{b[.],T}(\mathbf{w}^{b[.]}, \varepsilon_c, \theta^{t-1}) = \sum_{\mathbf{x}_i \in \mathcal{A}} \sum_{c \in \mathcal{C}} \sum_{c^* \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1}) \ln(f_c(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]}, \varepsilon_c))$$

  Cette partie est similaire au modèle BeSOM (chapitre 2), où le paramètre ϵ dépend seulement de la cellule c. La maximisation de la fonction est aussi réalisée en deux étapes. Une première maximisation par rapport à $w_c^{b[k]}$. Celle-ci correspond

au calcul du centre médian (Annexe A). Puis on effectue une maximisation par rapport à ε_c en résolvant l'équation $\frac{\partial Q_1^{b[.],T}(\mathbf{w}^{b[.]},\varepsilon_c,\theta^{t-1})}{\partial \varepsilon_c} = 0$. Ainsi les expressions qui permettent de maximiser cette partie de la fonction, connaissant les paramètres à l'instant $t-1$ sont définies par :

$$w_c^{b[k]} = \begin{cases} 0 & \text{si } \left[\sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i, \theta^{t-1})(1 - x_i^{b[k]})\right] \geq \\ & \left[\sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i, \theta^{t-1}) x_i^{b[k]}\right] \\ 1 & \text{sinon} \end{cases} \tag{3.11}$$

$$\varepsilon_c = \frac{\sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i, \theta^{t-1}) \mathcal{H}(\mathbf{x}_i^{b[.]}, \mathbf{w}_c^{b[.]})}{\sum_{\mathbf{x}_i \in \mathcal{A}} m p(c/\mathbf{x}_i, \theta^{t-1})} \tag{3.12}$$

où
$p(c^*/\mathbf{x}_i, \theta^{t-1}) = \sum_{c \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1})$
et
$p(c/\mathbf{x}_i, \theta^{t-1}) = \sum_{c^* \in \mathcal{C}} p(c, c^*/\mathbf{x}_i, \theta^{t-1})$.

Il est clair que le modèle PrMTM est la génération des modèles BeSOM et PrSOM (chapitre 2). Le premier terme $Q_1^{r[.],T}(\mathbf{w}^{r[.]}, \sigma_c, \theta^{t-1})$ correspond à la fonction objective de la version probabiliste PrSOM dédiée aux données continues (section 2.1). Le deuxième terme $Q_1^{b[.],T}(\mathbf{w}^{b[.]}, \varepsilon_c, \theta^{t-1})$ correspond à la fonction objective du modèle BeSOM-ε_c (chapitre 2, eq.2.11).

L'algorithme d'apprentissage PrMTM est une autre application de l'algorithme EM aux modèles des cartes auto-organisatrices probabilistes. En supposant que les probabilités conditionnelles $p(c^*/c)$ sont fixées pour une valeur donnée de T, l'algorithme est défini d'une manière itérative où les paramètres à l'itération $t+1$ sont calculés en fonction des paramètres estimés à l'itération t. L'algorithme PrMTM pour un paramètre T fixé se présente de la manière suivante :

1) **Initialisation** ($t = 0$)
   - détermination de l'ensemble des paramètres initiaux (θ^0) et du nombre d'itérations $iter$.
2) **Itération de base à T constant** ($t \geq 1$)

   - Calcul de l'ensemble des paramètres θ^{t+1} à partir de l'ensemble des paramètres déjà calculés θ^t en appliquant les formules (3.8),(3.9), (3.10),(4.11) and (4.12).

3) **Répéter** l'itération de base tant que $t < iter$.

Nous avons présenté le principe de l'algorithme d'apprentissage PrMTM permettant d'estimer les paramètres maximisant la fonction vraisemblance pour un paramètre T fixé. Le paramètre T permet de contrôler la taille du voisinage d'influence d'une cellule sur la carte, celle-ci décroît avec le paramètre T (fig.3.1). De la même manière et par analogie avec l'algorithme des cartes topologiques, on peut faire décroître la valeur de T entre deux valeurs T_{max} et T_{min}. Pour chaque valeur de T, on obtient une fonction de vraisemblance V^T qui varie avec T. Nous pouvons observer deux étapes dans le fonctionnement de l'algorithme.

- La première étape correspond aux grandes valeurs de T. Dans ce cas, le voisinage d'influence d'une cellule c de la carte est grand et correspond aux valeurs "significatives" de $K^T(\delta(c, r))$. Les formules, (3.8),(3.9), (3.10),(4.11) et (4.12) ont tendance, dans ce cas, à faire participer un très grand nombre d'observations à l'estimation des paramètres du modèle PrMTM.

- La deuxième étape correspond aux petites valeurs de T. Le nombre d'observations intervenant dans les formules (3.8),(3.9), (3.10),(4.11) et (4.12) est alors restreint. L'adaptation est très locale.

Si on utilise cette décroissance de T, l'algorithme d'apprentissage de PrMTM se présente de la manière suivante :

1) **Phase d'initialisation** ($t = 0$)

   - Choisir T_{max}, T_{min} et N_{iter}. Effectuer l'algorithme d'apprentissage PrMTM pour la valeur de T constante égale à T_{max}.

2) **Etape itérative**
   - L'ensemble des paramètres θ^t de l'étape précédente est connu. Calculer la nouvelle valeur de T en appliquant la formule suivante :

$$T = T_{max} \left(\frac{T_{min}}{T_{max}} \right)^{\frac{t}{iter-1}}$$

   - Pour cette valeur du paramètre T, calculer l'itération de base définie dans l'algorithme indiqué précédemment pour un paramètre fixe T.

3) **Répéter** l'étape itérative tant que $t \leq N_{iter}$

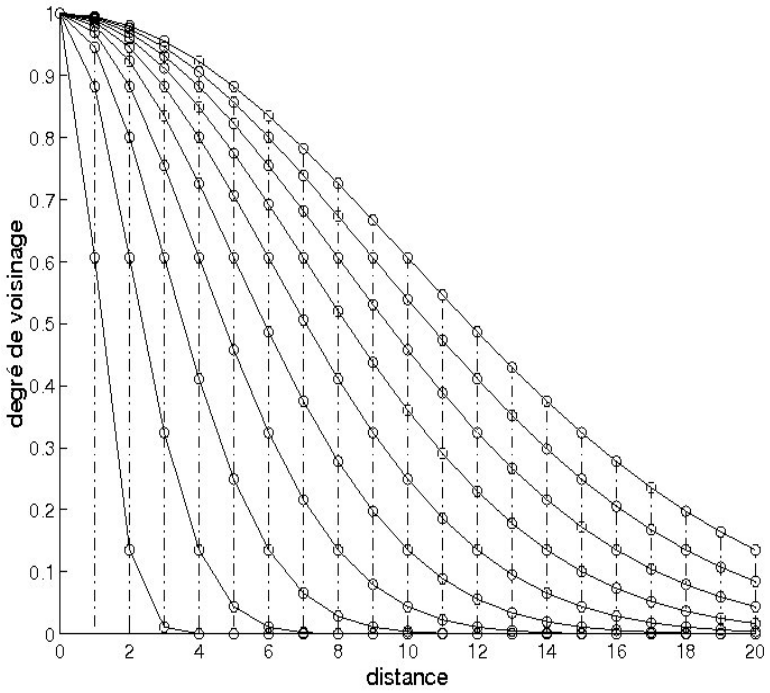


FIG. 3.1 – Ce graphique nous indique le degré de voisinage par rapport à la distance entre les cellules.

3.3 PrMTM et l'algorithme traditionnel des cartes topologiques

Il existe dans la littérature un modèle déterministe à base des cartes topologiques dédié aux données binaires et mixtes utilisant une fonction de coût similaire à l'algorithme traditionnel de Kohonen (MTM : Mixed Topological Map), [98, 105]. Etant donné que pour des vecteurs à composantes binaires la distance Euclidienne n'est que la distance de Hamming $\mathcal{H}$, alors la distance Euclidienne pour les données mixtes peut-être réécrite comme suit :

$$||\mathbf{x} - \mathbf{w}_c||^2 = ||\mathbf{x}^{r[.]} - \mathbf{w}_c^{r[.]}||^2 + \mathcal{H}(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]}).$$

Utilisant cette expression, la fonction de coût de l'algorithme classique de Kohonen peut être exprimée comme suit :

$$\begin{aligned} \mathcal{G}(\phi, \mathcal{W}) &= \sum_{\mathbf{x}_i \in A} \sum_{r \in C} \mathcal{K}^T(\delta(r, \phi(\mathbf{x}_i)))||\mathbf{x}_i^{r[.]} - \mathbf{w}_r^{r[.]}||^2 \\ &\quad + \sum_{\mathbf{x}_i \in A} \sum_{r \in C} \mathcal{K}^T(\delta(r, \phi(\mathbf{x}_i)))\mathcal{H}(\mathbf{x}_i^{b[.]}, \mathbf{w}_r^{b[.]}) \end{aligned} \tag{3.13}$$

où ϕ affecte chaque observation $\mathbf{x}$ à une seule cellule de $\mathcal{C}$. Le premier terme correspond à la fonction de coût classique utilisée par l'algorithme batch de Kohonen, et le deuxième terme représente la fonction de coût utilisée dans le modèle Binbatch dédié uniquement aux données binaires (section 3.3) [104]. La fonction de coût (eq.3.13), est minimisée en utilisant un processus itératif à deux étapes.

1) **L'étape d'affectation** qui utilise la fonction suivante :

$$\forall \mathbf{x},\ \phi(\mathbf{x}) = \arg\min_c \left(||\mathbf{x}^{r[.]} - \mathbf{w}_c^{r[.]}||^2 + \mathcal{H}(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]})\right)$$

2) **L'étape de quantification**. C'est facile d'observer que la minimisation de ces deux termes séparément nous permet de définir :
   - La partie quantitative $\mathbf{w}_c^{r[.]}$ du vecteur référent $\mathbf{w}_c$ est calculée comme suit :

$$\mathbf{w}_c^{r[.]} = \frac{\sum_{\mathbf{x}_i \in \mathcal{A}} \mathcal{K}(\delta(c, \phi(\mathbf{x}_i)))\mathbf{x}_i^{r[.]}}{\sum_{\mathbf{x}_i \in \mathcal{A}} \mathcal{K}(\delta(c, \phi(\mathbf{x}_i)))},$$

   - La partie binaire $\mathbf{w}_c^{b[.]}$ du vecteur référent $\mathbf{w}_c$ est définie comme le centre médian de la partie binaire des observations $\mathbf{x}_i^{b[.]} \in \mathcal{A}$ pondérées par $\mathcal{K}(\delta(c, \phi(\mathbf{x}_i)))$.

Chaque composante du vecteur $\mathbf{w}_c^{b[.]} = (w_c^{b[1]}, ..., w_c^{b[k]}, ..., w_c^{b[m]})$ est calculée de la manière suivante :

$$w_c^{b[k]} = \begin{cases} 0 & \text{si } \left[\sum_{\mathbf{x}_i \in \mathcal{A}} \mathcal{K}(\delta(c, \phi(\mathbf{x}_i)))(1 - x_i^{b[k]})\right] \geq \\ & \left[\sum_{\mathbf{x}_i \in \mathcal{A}} \mathcal{K}(\delta(c, \phi(\mathbf{x}_i)))x_i^{b[k]}\right] \\ 1 & \text{sinon} \end{cases},$$

L'analyse de ces formules indique que la mise à jour pour le centre médian $\mathbf{w}_c^{b[.]}$ et la partie réelle $\mathbf{w}_c^{r[.]}$ de notre modèle PrMTM coïncide avec le modèle BinBatch (section 3.3) et le modèle batch de Kohonen pour lesquels chaque observation $\mathbf{x}$ est pondérée proportionnellement par la fonction de voisinage ou une probabilité centrée sur le prototype gagnant.

Dans les deux modèles (PrMTM et MTM) nous minimisons l'inertie des observations $\mathbf{x}$ dans l'espace réel et binaire ($\Re^n \times \beta^m$). Dans le modèle probabiliste PrMTM, la définition du gagnant est différente de celle du modèle déterministe MTM. L'affectation est effectuée à la fin de l'algorithme d'apprentissage. Notons également qu'il existe un lien fort entre le modèle déterministe des cartes topologiques dédiées aux données mixtes (MTM : Mixed Topological Map), et le modèle de mélange PrMTM. A chaque pas d'apprentissage nous calculons la probabilité a posteriori $p(c/\mathbf{x})$ qui est utilisée pour pondérer l'observation $\mathbf{x}$. Dans le modèle MTM, l'affectation est simplifiée en minimisant $||\mathbf{x}^{r[.]} - \mathbf{w}_c^{r[.]}||^2 + \mathcal{H}(\mathbf{x}^{b[.]}, \mathbf{w}_c^{b[.]})$.

Ainsi, si nous supposons que ε et σ représentent le même paramètre pour toutes les cellules et pour toutes les variables (binaires et quantitatives), et si les probabilités a priori $p(c^*) = \frac{1}{K}$ sont égales, alors une distance faible implique une probabilité $p(\mathbf{x}/c) = p(\mathbf{x}^{r[.]}/c) \times p(\mathbf{x}^{b[.]}/c)$ élevée. Nous déduisons donc, que les probabilités a posteriori $p(c/\mathbf{x})$ et $p(c^*/\mathbf{x})$ deviennent élevées aussi. Celles-ci sont écrites de la manière suivante :

$$p(c/\mathbf{x}) = \frac{p(\mathbf{x}^{r[.]}/c) \times p(\mathbf{x}^{b[.]}/c) \sum_{c^* \in C} \mathcal{K}^T(\delta(c, c^*))}{KT_{c^*} p(\mathbf{x})},$$

$$p(c^*/\mathbf{x}) = \frac{\sum_{c \in C} \mathcal{K}^T(\delta(c, c^*)) p(\mathbf{x}^{r[.]}/c) \times p(\mathbf{x}^{b[.]}/c)}{KT_{c^*} p(\mathbf{x})},$$

Dans ces conditions, la maximisation de la fonction de coût $Q^T(\theta^t, \theta^{t-1})$ est la même que la maximisation de la fonction de coût $Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1})$ qui est décomposée en deux termes (eq. 3.7) et qui dépendent seulement de $\theta^c = (\mathbf{w}_c, \varepsilon, \sigma)$, où la probabilité ε et l'écart σ dépendent seulement de la carte. Par conséquent, maximiser $Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1})$

par rapport à ε et σ nécessite le calcul de la dérivée simple du $Q_1^{b[.],T}(\mathbf{w}^{b[.]}, \varepsilon, \theta^{t-1})$ par rapport à ε et $Q_1^{r[.],T}(\mathbf{w}^{r[.]}, \sigma_c, \theta^{t-1})$ par rapport à σ. Par contre, maximiser $Q_1^T(\theta^{\mathcal{C}}, \theta^{t-1})$ par rapport à $\mathbf{w}_c$ nécessite la minimisation de la fonction de coût simplifiée $G(\mathbf{w})$ qui est définie par :

$$\begin{aligned} G(\mathbf{w}) &= \sum_{c \in \mathcal{C}} \sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i, \theta^{t-1}) \mathcal{H}(\mathbf{x}_i^{b[.]}, \mathbf{w}_c^{b[.]}) \\ &\quad + \sum_{c \in \mathcal{C}} \sum_{\mathbf{x}_i \in \mathcal{A}} p(c/\mathbf{x}_i, \theta^{t-1}) ||\mathbf{x}_i^{r[.]} - \mathbf{w}_c^{r[.]}||^2 \end{aligned} \tag{3.14}$$

On déduit, ainsi, que le modèle MTM a tendance à rendre toutes les cellules équivalentes. La détermination de la cellule responsable de la génération c^* avec une fonction d'affectation, nous permet de déduire que la fonction de coût $\mathcal{G}(\phi, \mathbf{w})$ (eq.3.13) n'est qu'une simplification de la fonction de coût $G(\mathbf{w})$ (eq.3.14). Il est clair que le modèle PrMTM probabiliste fournit plus d'information que le modèle déterministe MTM.

3.4 Résultats expérimentaux

3.4.1 Jeux de données artificiel

Nous illustrons dans cette partie l'application du modèle PrMTM sur une base de données artificielles composée de 1500 observations bi-dimensionelles générées sous forme d'une structure de deux spirales. Dans cette base, la variable binaire dénote l'appartenance d'un point à l'une des deux spirales. L'objectif de cet exemple est de montrer que le modèle permet de construire des cartes auto-organisées en utilisant les deux types de variables.

Pour illustrer la phase d'apprentissage de notre modèle, nous avons entrainé trois cartes de dimension différentes (1×5, 1×10 et 10×10). Comme attendu, les résultats de la carte prennent en compte les étiquettes et la structure spatiale des données. La figure 3.2 permet de s'assurer que la composante qualitative codée en binaire est cohérente avec les composantes quantitatives qui représentent les coordonnées des points.

Nous observons sur cet exemple "jouet" que notre modèle de cartes probabilistes respecte l'ordre topologique des données malgré le mélange de lois de probabilités. En

effet, avec la carte 10×10 les prototypes ont une petite probabilité ε_c d'être différents de l'étiquette de la classe. Par contre les ficelles 1×5 et 1×10 suivent la structure de la carte, mais avec une confiance faible exprimée par des probabilités fortes (Le rayon du cercle est proportionnel à la probabilité ε).

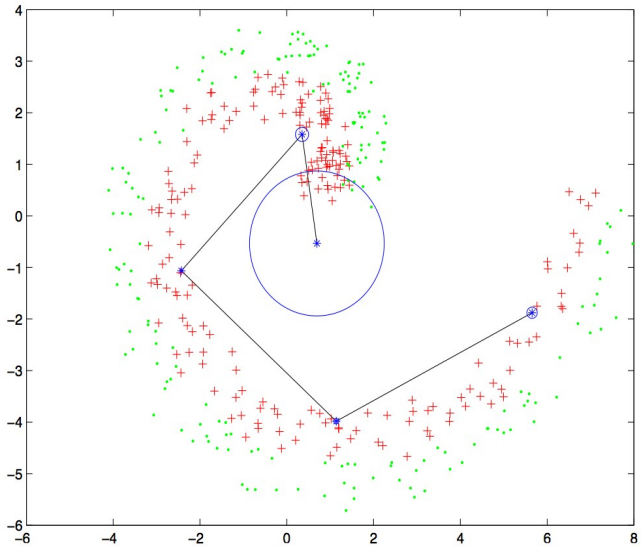
3.4.2 La base de données "Cleve"

Dans cette partie, nous illustrons la convergence de l'algorithme PrMTM sur la base médicale *cleve*. Cette base avait été fournie par le Dr. Detrano en 1988. Elle décrit la maladie du coeur traitée dans la clinique de Cleveland. L'ensemble des données contient 303 patients où chacun est décrit par 6 variables quantitatives et 7 qualitatives. Les patients sont classés en deux classes, chacune représente la présence de la maladie ou son absence. En utilisant un codage disjonctif binaire on obtient 17 variables binaires.

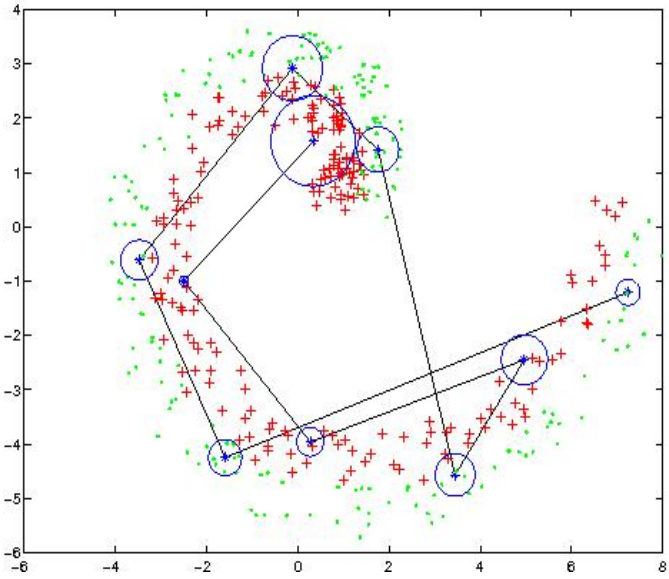
L'apprentissage d'une carte de dimension 13×7, fournit pour chaque cellule des vecteurs référents avec deux parties $(\mathbf{w}_c^{r[.]}, \mathbf{w}_c^{b[.]})$ associées respectivement à l'écart-type σ_c pour la partie quantitative et la probabilité ε_c d'être différent du prototype pour la partie binaire. La figure 3.3, illustre la maximisation de la fonction de coût $Q^T(\theta, \theta^t)$.

Notre modèle PrMTM nous a permis de disposer des mêmes visualisations que les cartes topologiques traditionnelles. Nous rappelons que le modèle PrMTM utilise une affectation probabiliste pour projeter les observations sur la carte. La figure 3.4 illustre la variation de la partie quantitative correspondant à toutes les observations $\mathbf{x}^{r[.]}$ captées par chaque cellule. On observe une auto-organisation de la partie quantitative malgré la présence de la partie binaire lors de la phase d'apprentissage.

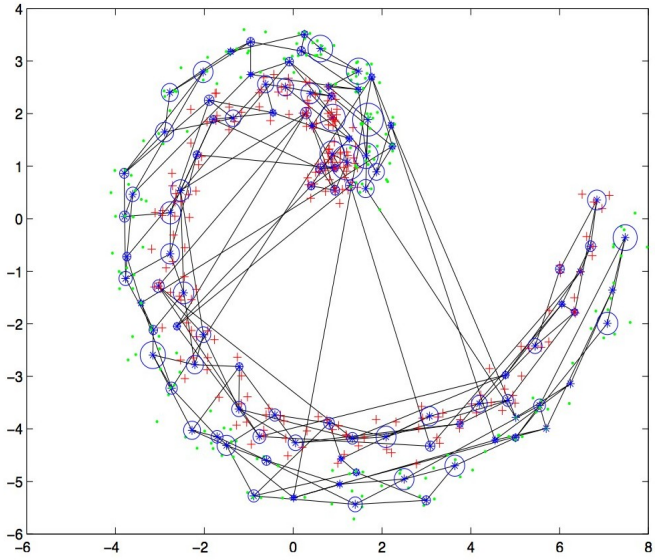
La figure 3.5 affiche la probabilité ε_c d'être différent de la partie binaire $\mathbf{w}^{b[.]}$ indiquée par la partie binaire (figure 3.6). La nuance de gris de chaque prototype est proportionnelle à la probabilité d'être différent du prototype binaire estimé. Cette information est très importante pour l'interprétation des résultats. La probabilité ε indique la confiance qu'on peut faire aux variables binaires et par conséquent aux variables qualitatives. Lorsque la probabilité est très forte, l'expert peut décider d'ignorer l'interprétation des variables binaires correspondantes.



1×5



1×10



10×10

FIG. 3.2 – Carte PrMTM (1×5, 1×10, 10×10). Nous visualisons la partie des données quantitatives $\mathbf{w}_c^{r[1,2]}$ et la probabilité ε_c d'être différent de l'étiquette $\mathbf{w}_c^{b[3]}$. Le rayon du cercle est proportionnel à la probabilité ε_c.

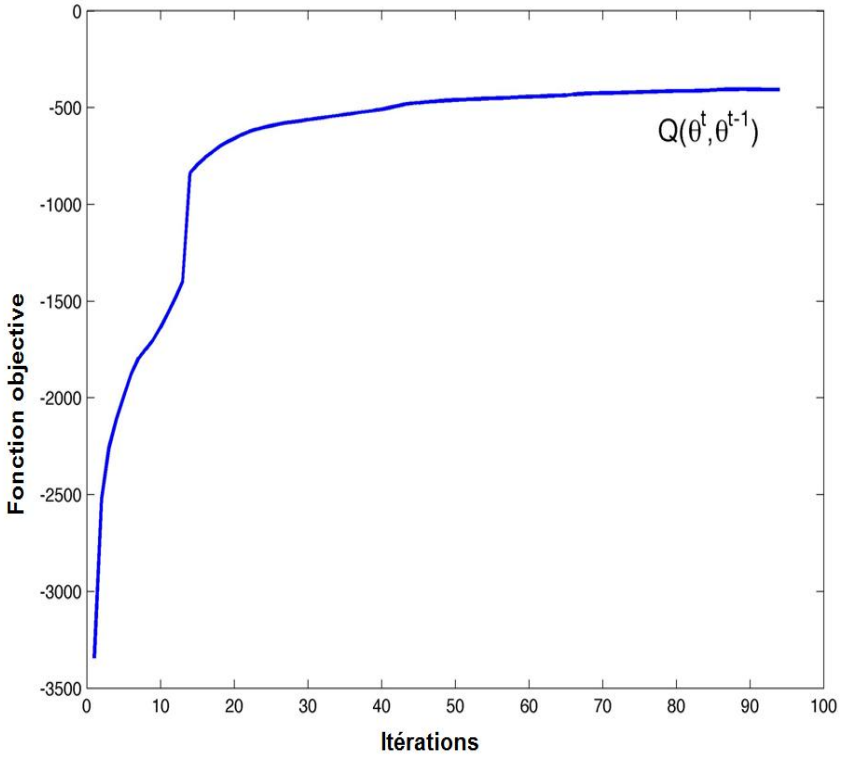


FIG. 3.3 – Evolution de la fonction de coût $Q^T(\theta, \theta^t)$ selon le nombre d'itérations.

3.4.3 Autre données réelles

Dans cette section nous montrons les contributions de notre modèle par rapport à l'algorithme déterministe de classification topologique MTM. Pour la suite on utilise deux autres bases obtenues du répertoire UCI [24]. Le tableau 3.1, fournit une courte description des bases de données utilisées.

Base de données	dim. qualit	dim. quantitative	# obs	# classes
Cleve	7	6	303	2
Credit	6	9	666	2
Thyroid	12	7	3163	2

TAB. 3.1 – Jeux de données utilisées pour l'évaluation. # obs : nombre d'observation ; # classes : nombre de classes.

Nous avons comparé notre modèle PrMTM avec l'algorithme classique déterministe MTM. Dans ces expérimentations, la comparaison des différents résultats est mesurée

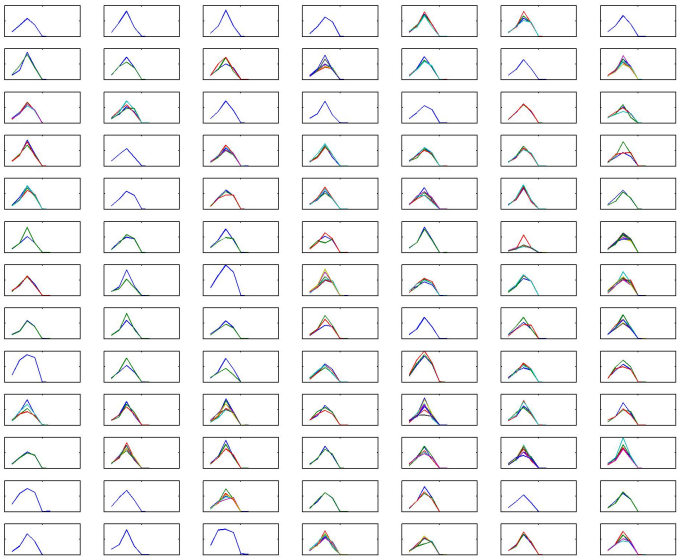


FIG. 3.4 – La carte PrMTM pour la base de données *Cleve*. Chaque cellule indique la partie quantitative des observations captées par chaque cellule.

à l'aide du taux de pureté en utilisant l'étiquette connue de chaque observation. La comparaison est réalisée en calculant la moyenne des puretés sur 50 expériences. Le tableau 3.2 montre les performances atteintes avec notre modèle PrMTM et le modèle MTM. Nous observons une amélioration des puretés sur toutes les bases.

Pureté : %	MTM	PrMTM
Cleve (13×7)	86.78 ± 2.62	87.25 ± 1.95
Credit (13×10)	81.60 ± 2.46	81.93 ± 1.96
Thyroid (21×14)	95.85 ± 1.06	96.22 ± 1.09

TAB. 3.2 – Comparaison entre MTM et PrMTM utilisant l'indice de pureté (moyenne± écart-type sur 50 expérimentations). MTM : Carte topologique classique dédiées aux données mixtes. PrMTM : carte topologique utilisant la loi Gaussienne et Bernoulli

En examinant le tableau 3.2, nous observons par exemple, pour la base *Cleve* une amélioration de la pureté de 86.78% à 87.25% en réduisant l'écart-type de 2.62 à 1.95. Pour la base de données *Credit* nous observons des résultats équivalents, mais en réduisant l'écart-type de 2.46 à 1.96. En ce qui concerne la base de données *Thyroid* on observe une amélioration de pureté avec une légère augmentation de l'écart-type.

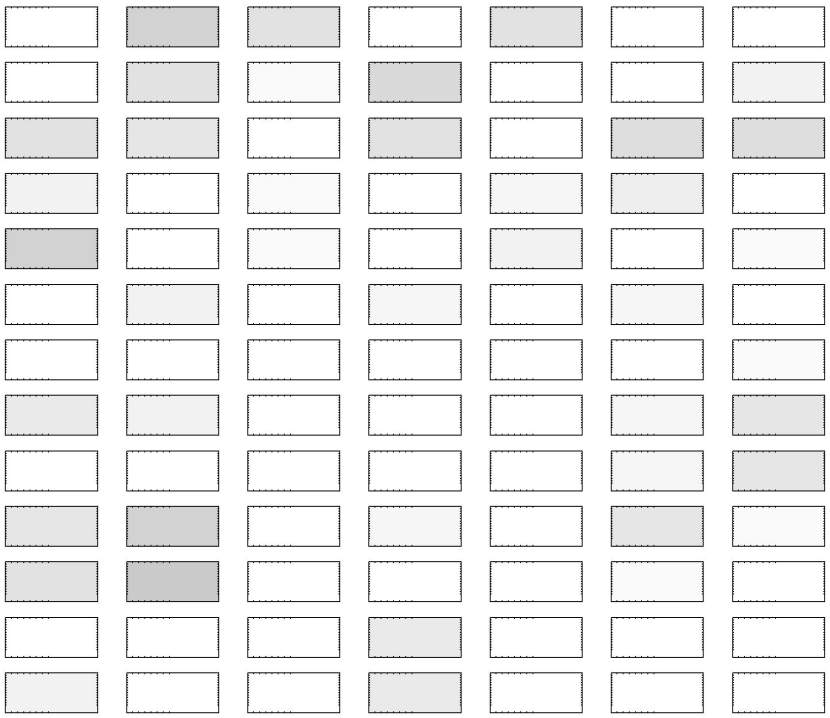


FIG. 3.5 – La carte PrMTM pour les de données *Cleve* de dimension 13×7. La carte indique la probabilité ε_c. Le niveau de gris de chaque cellule est proportionnel à la probabilité d'être différent du prototype binaire estimé $\mathbf{w}_c^{b[.]}$.

Taux de classification : %	MTM	PrMTM-ϵ_c	PrMTM
Cleve (13×7)	72.78	75.26	74.93
Credit (13×10)	62.76	64.58	63.34
Thyroid (21×14)	82.75	85.41	84.69

TAB. 3.3 – Comparaison entre MTM, PrMTM et PrMTM-ϵ_c utilisant la technique de la validation croisée. On indique le taux de classification pour chaque base. MTM : Carte topologique classique dédiées aux données mixtes. PrMTM : carte topologique utilisant la loi Gaussienne et Bernoulli (où la probabilité est calculée pour chaque cellule de la carte). PrMTM-ϵ_c : carte topologique utilisant la loi Gaussienne et Bernoulli (où le vecteur probabilité ϵ_c dépend de la cellule c et de la variable, $\epsilon_c = (\varepsilon_c^1, ..., \varepsilon_c^k, ..., \varepsilon_c^n)$)

Pour valider notre approche, nous avons utilisé la validation croisée. Par conséquent, nous avons divisé les bases en 3 sous-ensembles (un tiers de la base est utilisé pour la

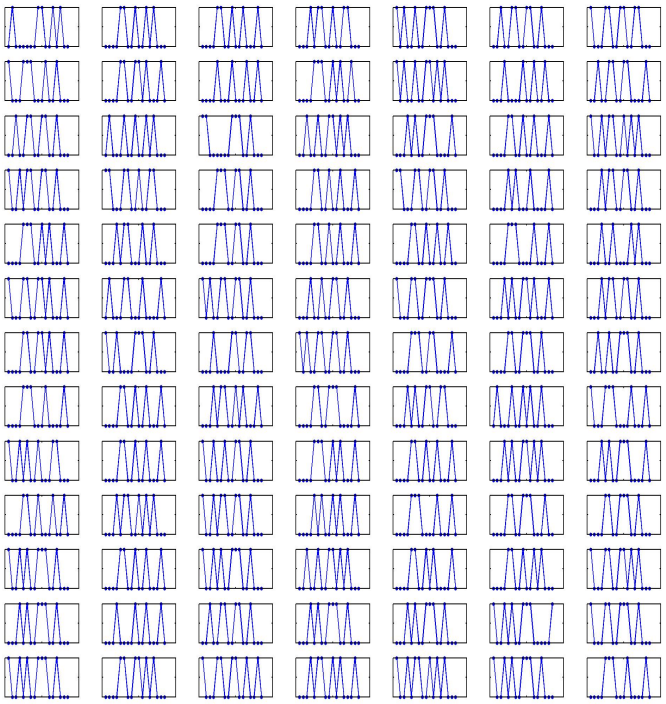


FIG. 3.6 – La carte PrMTM pour les de données *Cleve* de dimension 13×7. La carte indique les profils de la partie binaire ("1/0"qualitative).

validation et deux tiers pour l'apprentissage). Ainsi, 15 expériences sont réalisées pour chaque base. Le tableau 3.3 indique la moyenne calculée avec 15 expériences. Pour le même objectif de comparaison, nous avons enrichi le modèle PrMTM avec l'estimation d'un vecteur de probabilités par cellule au lieu d'une valeur par cellule. Ce modèle fait référence au modèle modèle BeSOM-ϵ_c dédié aux données binaires où la probabilité dépend à la fois de la cellule et de la variable, $\epsilon_c = (\varepsilon_c^1, ..., \varepsilon_c^k, ..., \varepsilon_c^n)$. Nous appelons par la suite ce modèle PrMTM-ϵ_c.

Nous observons clairement que le modèle PrMTM fournit de meilleurs résultats que le modèle déterministe. Ceci est rassurant puisque le modèle PrMTM est plus riche en information. Il est claire aussi que lorsqu'on estime un vecteur de probabilité pour la partie binaire "PrMTM-ϵ_c", les performances augmentent. Comme nous l'avons signalé dans les sections précédentes, le modèle PrMTM fournit plus d'informations que le modèle déterministe MTM. Pour le modèle proposé, la probabilité ε_c et l'écart-type

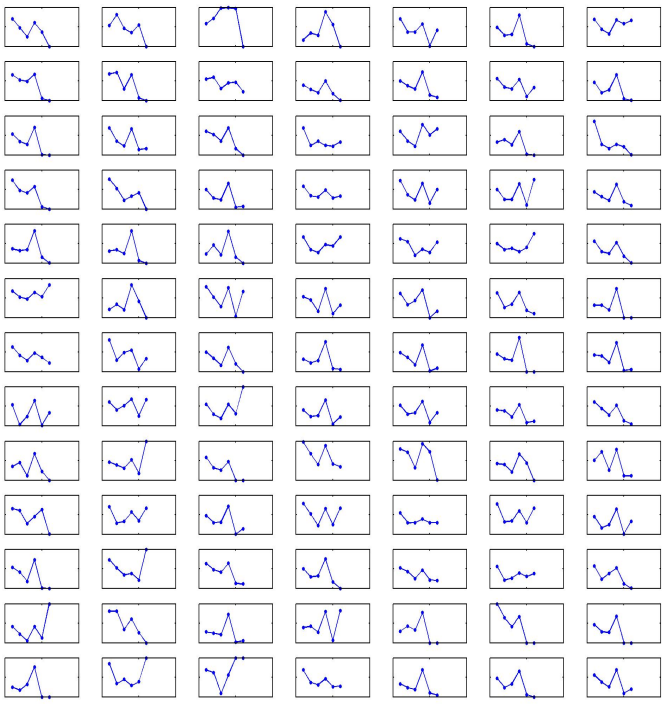


FIG. 3.7 – La carte PrMTM pour les de données *Cleve* de dimension 13×7. La carte indique les profils de la partie réelle de la carte.

σ_c peuvent être utilisés pour la sélection des modèles. Supposant que la probabilité et l'écart-type dépendent des prototypes et des variables, on estime pour les deux distributions, le vecteur d'écart-type $\sigma_c = (\sigma_c^1, \sigma_c^2, ..., \sigma_c^n)$ et le vecteur de probabilités $\epsilon_c = (\varepsilon_c^1, \varepsilon_c^2, ..., \varepsilon_c^m)$. Ainsi ce modèle peut-être utilisé pour la sélection non-supervisée des variables. Il n'est pas évident de comparer notre modèle avec un algorithme qui n'utilise pas une architecture similaire. Nous rappelons que le modèle PrMTM, basé sur l'architecture SOM, fournit plus de clusters que les modèles hiérarchiques ou le K-means dédié aux données mixtes [1].

3.5 Conclusion

Ce chapitre présente une approche probabiliste dédiée aux données mixtes (quantitatives et binaires) qui utilise l'algorithme EM pour maximiser la vraisemblance de données afin d'estimer les paramètres d'un modèle de mélange des lois de Bernoulli et Gausiennes. Cet algorithme a l'avantage de fournir des prototypes qui sont de la même nature que les données initiales. Nous avons montré que l'algorithme PrMTM nous fournit différentes informations qui peuvent être utilisées dans des applications pratiques. Dans le chapitre suivant, nous allons nous intéresser au même formalisme qui sera dédié aux données structurées en séquences.

Chapitre 4

Classification topologique Markovienne

4.1 Introduction

Les données séquentielles consistent à représenter des ensembles d'unités d'une longueur fixe ou variable et éventuellement d'autres caractéristiques intéressantes, tels que les comportements dynamiques et les contraintes de temps. Ces propriétés rendent les données séquentielles distinctes par rapport aux autres types de données que nous avons vu dans les chapitres précédents, et soulèvent de nombreuses nouvelles questions. Il est souvent impossible de les représenter comme des points dans un espace multidimensionnel et d'utiliser les algorithmes de classification existants. Notez que les données séquentielles peuvent générer des clusters inexistants, comme c'est indiqué dans les travaux de Lin et Chen [107]. Si on observe les points de la figure 4.1(a), en vérifiant s'ils forment une séquence, on constate que la partition obtenue n'est pas raisonnable parce que le point étiquetté avec 10, n'est pas en continuité avec les autres points étiquetés de 1 à 7. La nouvelle partition est représentée dans la figure 4.1(b), où le point 10 est groupé avec les points 8-9 et 11-16 pour former une séquence continue, même s'il est plus "proche", dans le sens de la distance euclidienne, du cluster formé par les points 1 à 7.

Les données séquentielles pourraient être générées à partir d'un grand nombre de sources telles que : le traitement de la parole, la fouille de texte, le séquençage de l'ADN, le diagnostic médical, l'analyse du stock du marché, les transactions des clients,

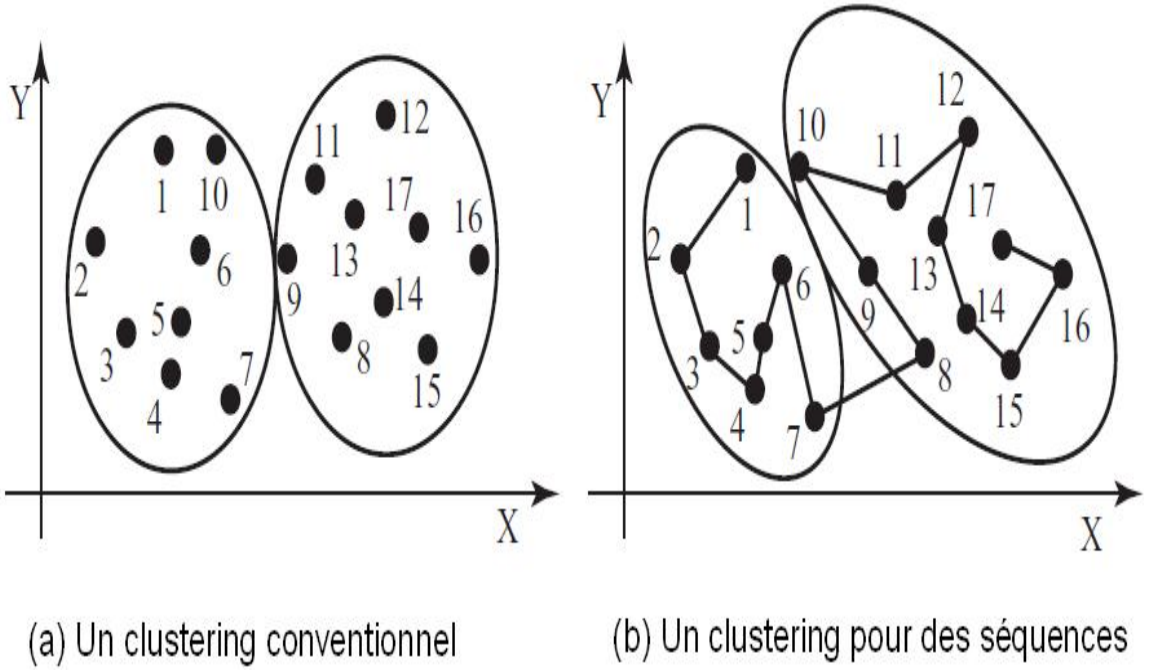


FIG. 4.1 – La différence entre un clustering conventionnel et un clustering pour les séquences ((figure prise de p.110 [107]).

la fouille de données web, l'analyse des capteurs en robotique. Elles peuvent être sous forme de données continues, telles que les grandes masses de données temporelles, ou composé de symboles non numériques, par exemple, les nucléotides pour les séquences ADN et les aminoacides pour les séquences des protéines. Ainsi depuis plusieurs années, les séquences temporelles et spatiales ont été le sujet de recherche dans plusieurs domaines tels que : la statistique, la reconnaissance des formes, le Web mining et la bioinformatique. La méthode la plus facile pour traiter les données séquentielles serait tout simplement d'ignorer l'aspect temporel ou séquentiel et de traiter les observations comme des données indépendantes ou i.i.d "independent and identically distributed". Pour beaucoup d'applications, l'hypothèse i.i.d rend les données plus pauvres en perdant l'information séquentielle. Souvent dans beaucoup d'applications le traitement est décomposé en deux étapes : la première est l'étape de classification ou de partitionnement des données avec l'hypothèse i.i.d. Dans la deuxième étape, le résultat de la classification est utilisé pour construire un modèle probabiliste en relaxant la contrainte i.i.d, et une des plus naturelles manières de faire cela est d'utiliser un modèle de Markov.

Les Chaines de Markov Cachées (HMMs) figurent parmi les meilleures approches adaptées aux traitements des séquences, du fait d'une part de leur capacité à traiter des

séquences de longueurs variables, et d'autre part de leur pouvoir à modéliser la dynamique d'un phénomène décrit par des suites d'événements. C'est pourquoi ces modèles ont été largement utilisés dans le domaine de la reconnaissance de la parole où les données se présentent de manière séquentielle. Ces chaînes de Markov Cachées (HMMs) proposent une solution à ce problème en introduisant, pour chaque état, un processus stochastique sous-jacent qui n'est pas connu (caché), mais pourrait être déduit par les observations qu'il génère. En fait, la modélisation graphique probabiliste motive différentes structures graphiques basées sur les HMMs [18, 17]. Une des variantes du modèle HMM est le modèle de Markov caché factoriel [71], dans lequel on a plusieurs chaînes de Markov indépendantes des variables latentes, où la distribution de la variable observable à un instant donné est conditionnée par les états de toutes les variables latentes correspondantes au même instant. On trouve aussi dans la littérature plusieurs méthodes hybrides, composées des HMM avec les réseaux de neurones et plus particulièrement les cartes auto-organisatrices [30, 117]. Evidemment, il existe plusieurs structures probabilistes possibles qui peuvent être construites par rapport aux besoins des applications particulières. Les modèles graphiques fournissent un formalisme général pour décrire et analyser de telles structures. Par conséquent, il est très important d'avoir des algorithmes capables de déduire à partir d'un ensemble de données de séquences non seulement la probabilité de distribution, mais aussi la structure topologique du modèle, c'est-à-dire, le nombre d'états et les transitions qui les interconnectent. Malheureusement, cette tâche est très difficile et on a que des solutions partielles [29, 27, 28]. Afin de surmonter les limites des HMMs, des travaux récents dans [27, 28] proposent un nouveau et un original paradigme, appelé *topological HMM*, qui manipule les noeuds du graphe associé au HMM et ses transitions dans un espace Euclidien. Cette approche modélise la structure locale d'un HMM et extrait leur forme en définissant une unité d'information comme une forme composée d'un groupe de symboles d'une séquence.

D'autres approches ont été proposées pour combiner les HMMs et les cartes auto-organisatrices (Sel-Organizing Map) pour obtenir des modèles hybrides qui contiennent le pouvoir de partitionnement du SOM et qui modélisent la notion de dynamique dans les séquences avec le HMM. Les cartes topologiques sont intéressantes de par leurs apports topologiques à la classification non supervisée et leurs capacités à résumer de manière simple un ensemble de données multi-dimensionnelles. Elles permettent d'une part de comprimer de grandes quantités de données en regroupant les individus similaires en classes, et d'autre part de projeter les classes obtenues de façon non linéaire sur une carte ou un graphe - donc d'effectuer une réduction de dimension -, permettant ainsi de visualiser la structure des données en deux dimensions tout en respectant

la topologie des données, c'est à dire de sorte que deux données proches dans l'espace multi-dimensionnel de départ aient des images proches sur la carte. Dans la plupart des modèles hybrides, les cartes topologiques sont utilisées comme un prétraitement pour la quantification vectorielle et les HMMs sont ensuite utilisés dans des processus de transformation plus avancés pour la prise en compte de la dynamique. Dans [146, 145], un vecteur d'éléments structurés en séquence est associé à un noeud (cellule) de la carte topologique en utilisant la DTW (Dynamic Time Warping). Il existe plusieurs travaux qui décrivent différentes méthodes et approches de combinaison [8, 58, 59]. Dans [59] les auteurs proposent un modèle original qui est le résultat de la collaboration de l'algorithme SOM et la théorie des HMMs. Le modèle de base consiste en un nouvel algorithme unifié/hybride SOM-HMM dans lequel chaque cellule de la carte modélise un HMM. Cependant, le processus d'organisation n'est pas intégré explicitement dans l'approche HMM.

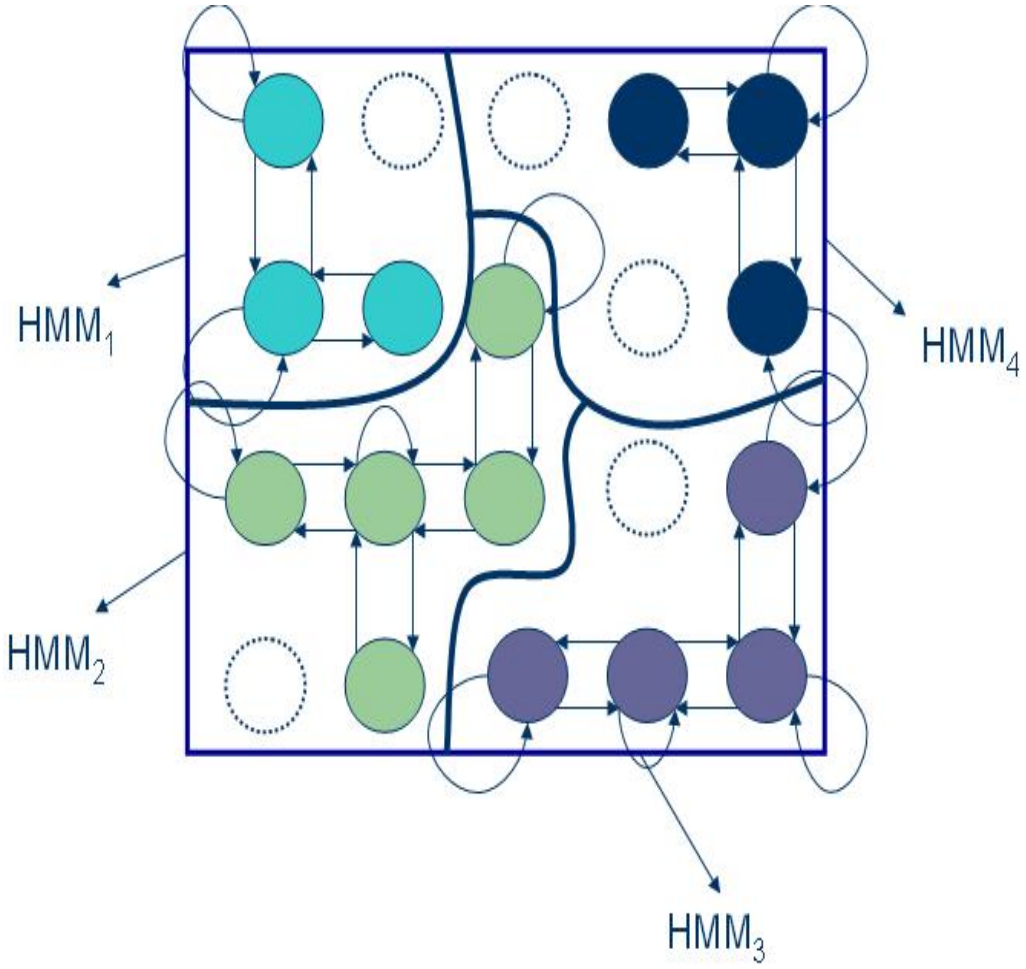
Dans ce chapitre, nous nous sommes intéressés à la problématique du traitement de données structurées en séquences, qu'elles soient de longueurs fixes ou variables en adoptant deux approches (figure 4.2) :

- La première est une approche locale qui consiste à voir la carte SOM comme un modèle de mélange de HMMs, où chaque HMM représente une région précise de la carte.

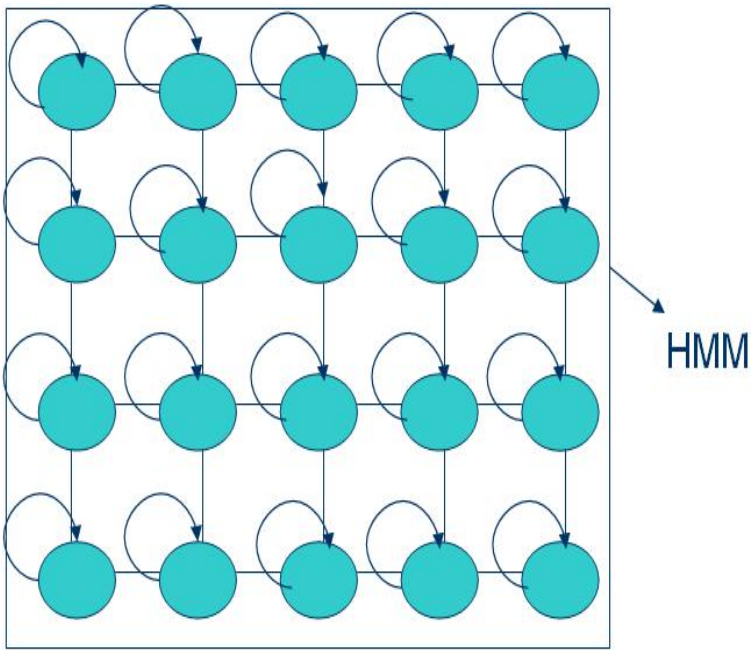
- La deuxième voie globale consiste à modéliser la carte topologique comme un seul modèle de mélange markovien HMM.

4.2 L'approche globale

L'objectif de cette approche est de construire un nouveau modèle pour automatiser et auto-organiser la construction d'un modèle statistique génératif d'un ensemble de données de séquences. Dans notre modèle global, on considère qu'on a une chaîne de Markov qui forme une grille. La génération d'une variable observable à un instant donné du temps est conditionnée par les états voisins au même instant du temps. Ainsi, une grande proximité implique une grande probabilité pour la contribution de la génération. Cette proximité est quantifiée en utilisant la fonction de voisinage. Le même principe est utilisé par l'algorithme de Kohonen pour les données i.i.d [98]. Mais dans notre cas on se concentre sur des observations non i.i.d.



(a) Approche locale



(b) Approche globale

FIG. 4.2 – Deux approches de modélisation des cartes topologiques et du modèle markovien.

4.2.1 Le Modèle de Markov auto-organisé

On suppose que l'architecture de HMM est un treillis $\mathcal{C}$, qui a une topologie discrète définie par un graphe non orienté. On notera le nombre des cellules (noeuds, états) de $\mathcal{C}$ par K. Pour chaque paire de cellules (c,r) dans le graphe, la distance $\delta(c,r)$ est définie comme la longueur de la plus courte chaîne qui lie les cellules r et c. L'architecture du modèle est inspirée de la classification topologique probabiliste des données i.i.d utilisant les cartes auto-organisatrices de Kohonen définies dans les chapitres 2 et 3 [6, 22, 106].

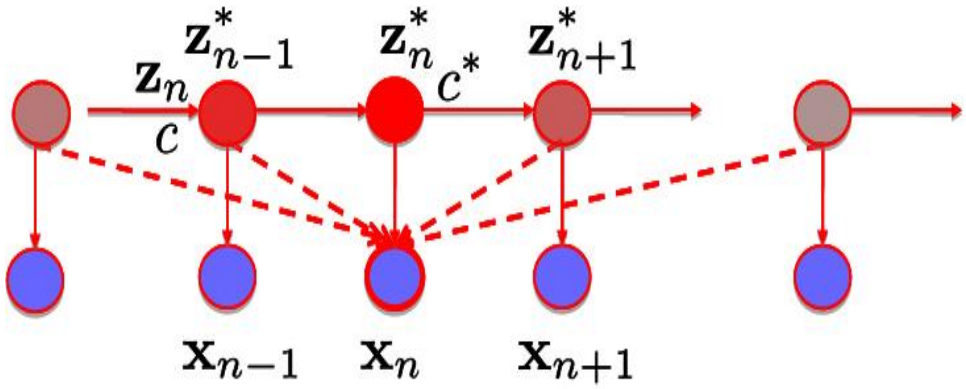


FIG. 4.3 – Un fragment de la représentation graphique de l'approche globale.

En s'inspirant des modèles des cartes topologiques probabilistes (chapitre 2 et 3), on suppose que chaque élément $\mathbf{x}_n$ d'une séquence d'observations $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n, ..., \mathbf{x}_N\}$ est généré par le processus suivant : on commence par associer à chaque cellule (état) $c \in \mathcal{C}$ une probabilité $p(\mathbf{x}_n/c)$ où $\mathbf{x}_n$ est un vecteur dans l'espace des données. Par la suite, on sélectionne une cellule c^* de la carte $\mathcal{C}$ selon une probabilité a priori $p(c^*)$. Pour chaque cellule c^*, on sélectionne une cellule $c \in \mathcal{C}$ selon la probabilité conditionnelle $p(c/c^*)$. Toutes les cellules $c \in \mathcal{C}$ et au même instant n contribuent à la génération d'un élément $\mathbf{x}_n$ avec $p(\mathbf{x}_n/c)$ selon la proximité à c^* décrite par la probabilité $p(c/c^*)$. Ainsi, une grande proximité à c^* implique une grande probabilité $p(c/c^*)$, et donc la contribution de l'état c à la génération de $\mathbf{x}_n$ est grande.

A la différence des modèles de mélanges présentés dans les chapitres précédents, nous avons décidé d'introduire deux variables binaires aléatoires comme variables cachées $\mathbf{z}_n$ et $\mathbf{z}^*_n$ de dimension K, dans lesquelles un élément particulier z_{nk} et z^*_{nk} est égal à 1

et tous les autres éléments sont égaux à 0. Les deux composantes z^*_{nk} et z_{nk} indiquent un couple d'états responsable de la génération d'un élément de l'observation. Utilisant cette notation on peut réécrire la probabilité $p(\mathbf{x}_n/c)$ comme suit :

$$p(\mathbf{x}_n/c) \equiv p(\mathbf{x}_n/z_{nc} = 1) \equiv p(\mathbf{x}_n/\mathbf{z}_n)$$

et

$$p(c/c^*) = p(z_{nc} = 1/z^*_{nc^*} = 1) \equiv p(z_{nc}/z^*_{nc^*}) \equiv p(\mathbf{z}_n/\mathbf{z}^*_n)$$

Pour introduire le processus d'auto-organisation dans l'apprentissage du modèle de mélange on suppose que $p(z_{nc}/z^*_{nc^*})$ peut être définie de la même manière que les modèles des cartes probabilistes (chapitre 2 et 3) :

$$p(z_{nc}/z^*_{nc^*}) = \frac{\mathcal{K}^T(\delta(c, c^*))}{\sum_{r\in\mathcal{C}} \mathcal{K}^T(\delta(r, c^*))},$$

où $\mathcal{K}^T$ est la fonction de voisinage qui dépend du paramètre T (appelé température) : $\mathcal{K}^T(\delta) = \mathcal{K}(\delta/T)$. Ainsi $\mathcal{K}$ définit pour chaque état de la chaîne de Markov $z^*_{nc^*}$ une région de voisinage dans le graphe $\mathcal{C}$. Le paramètre T permet de contrôler la taille du voisinage qui influence une cellule donnée de la carte $\mathcal{C}$. Comme dans le cas de l'algorithme de Kohonen pour les données i.i.d, la valeur de T varie entre deux valeurs T_{max} et T_{min}.

On note l'ensemble de toutes les variables cachées par $\mathbf{Z}^*$ et $\mathbf{Z}$, où chaque ligne $\mathbf{z}^*_n$ et $\mathbf{z}_n$ est associée à chaque élément de la séquence $\mathbf{x}_n$. Chaque observation de la séquence en X, est associée à un couple de variables cachées $\mathbf{Z}$ et $\mathbf{Z}^*$ responsables de la génération. On note par $\{\mathbf{X}, \mathbf{Z}, \mathbf{Z}^*\}$ l'ensemble complet des données, et on se réfère aux données observables $\mathbf{X}$ comme incomplètes. Ainsi, le modèle générateur d'une séquence est défini de la manière suivante :

$$p(\mathbf{X}; \theta) = \sum_{\mathbf{Z}^*} \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}, \mathbf{Z}^*; \theta) \tag{4.1}$$

Puisque la distribution $p(\mathbf{X}, \mathbf{Z}, \mathbf{Z}^*; \theta)$ ne peut pas se simplifier, on ne peut pas traiter chaque somme indépendamment. Une caractéristique importante pour les distributions des probabilités sur des variables multiples est celle de l'indépendance conditionnelle [108]. On suppose que la distribution conditionnelle de $\mathbf{X}$, sachant $\mathbf{Z}^*$ et $\mathbf{Z}$, ne dépend pas de la variable cachée $\mathbf{Z}^*$. Souvent cette hypothèse est utilisée pour les modèles graphiques, ainsi $p(\mathbf{X}/\mathbf{Z}, \mathbf{Z}^*) = p(\mathbf{X}/\mathbf{Z})$. Dans ce cas la distribution jointe des observations de la séquence est égale à :

$$p(\mathbf{X}, \mathbf{Z}^*, \mathbf{Z}) = p(\mathbf{Z}^*)p(\mathbf{Z}/\mathbf{Z}^*)p(\mathbf{X}/\mathbf{Z})$$

et on peut réécrire la distribution marginale comme :

$$p(\mathbf{X};\theta) = \sum_{\mathbf{Z}^*} p(\mathbf{Z}^*) \sum_{\mathbf{Z}} p(\mathbf{Z}/\mathbf{Z}^*) p(\mathbf{X}/\mathbf{Z}) \tag{4.2}$$

avec

$$p(\mathbf{X}/\mathbf{Z}^*) = \sum_{\mathbf{Z}} p(\mathbf{Z}/\mathbf{Z}^*) p(\mathbf{X}/\mathbf{Z}) \tag{4.3}$$

4.2.1.1 Paramètres du modèle topologique markovien

Considérant que la carte $\mathcal{C}$ représente un modèle de Markov, ainsi la distribution à l'état $\mathbf{z}_n^*$ dépend de l'état de la variable latente précédente $\mathbf{z}_{n-1}^*$. Cette dépendance est représentée avec la probabilité conditionnelle $p(\mathbf{z}_n^*|\mathbf{z}_{n-1}^*)$. Puisque les variables latentes sont des variables binaires de dimension K, cette distribution conditionnelle correspond à une table de probabilité qu'on note par $\mathbf{A}$. Les éléments de $\mathbf{A}$ sont connus comme des probabilités de transition notés par

$$A_{jk} = p(z_{nk}^* = 1/z_{n-1,j}^* = 1) \text{ avec } \sum_k A_{jk} = 1$$

La matrice $\mathbf{A}$ a au maximum $K(K-1)$ paramètres indépendants. Dans notre cas le nombre de transitions est limité par les noeuds de la carte. On peut écrire la distribution conditionnelle explicitement sous cette forme

$$p(\mathbf{z}_n^*/\mathbf{z}_{n-1}^*, \mathbf{A}) = \sum_{k=1}^{K} \sum_{j=1}^{K} A_{jk}^{z_{n-1,j}^* z_{nk}^*}$$

Toutes les distributions conditionnelles qui manipulent les variables cachées partagent les mêmes paramètres $\mathbf{A}$. L'état initial $\mathbf{z}_1^*$ est un cas particulier puisqu'il n'a pas de cellule parente, et ainsi il a une distribution marginale $p(\mathbf{z}_1^*)$ représentée par un vecteur de probabilités π avec les éléments $\pi_k = p(\mathbf{z}_{1k}^* = 1)$, ainsi que $p(\mathbf{z}_1^*|\pi) = \prod_{k=1}^{K} \pi^{z_{1k}^*}$, où $\sum_k \pi_k = 1$.

Les paramètres du modèle sont complétés en définissant les distributions conditionnelles des varaibles observées $p(\mathbf{x}_n/\mathbf{z}_n;\phi)$, où ϕ est un ensemble de paramètres qui définissent la distribution qui est connue comme des probabilités d'émission dans le modèle HMM. Puisque $\mathbf{x}_n$ est observable, la distribution $p(\mathbf{x}_n/\mathbf{z}_n,\phi)$ consiste, pour une valeur donnée de ϕ, d'un vecteur de K composantes qui correspondent aux K états possibles du vecteur binaire $\mathbf{z}_n$. On peut représenter les probabilités d'émission

sous la forme suivantes :

$$p(\mathbf{x}_n/\mathbf{z}_n;\phi) = \prod_{k=1}^{K} p(\mathbf{x}_n;\phi_k)^{z_{nk}}$$

La probabilité jointe des variables observables et les deux variables latentes $\mathbf{Z}$ et $\mathbf{Z}^*$ est exprimée par :

$$p(\mathbf{X},\mathbf{Z}^*,\mathbf{Z};\theta) = p(\mathbf{Z}^*;\mathbf{A}) \times p(\mathbf{Z}/\mathbf{Z}^*) \times p(\mathbf{X}/\mathbf{Z};\phi)$$

$$\begin{aligned} p(\mathbf{X},\mathbf{Z}^*,\mathbf{Z};\theta) &= \left[p(\mathbf{z}_1^*|\pi)\prod_{n=2}^{N} p(\mathbf{z}_n^*/\mathbf{z}_{n-1}^*;\mathbf{A})\right] \\ &\times \left[\prod_{i=1}^{N} p(\mathbf{z}_i/\mathbf{z}_i^*)\right] \\ &\times \left[\prod_{m=1}^{N} p(\mathbf{x}_m/\mathbf{z}_m;\phi)\right] \end{aligned} \tag{4.4}$$

où $\theta = \{\pi, \mathbf{A}, \phi\}$ décrit l'ensemble des paramètres qui manipulent le modèle. Il n'est pas évident de maximiser la fonction de vraisemblance, à cause de la complexité de l'expression. C'est pour cela qu'on utilise l'algorithme EM pour trouver les paramètres qui maximisent la fonction de vraisemblance. L'algorithme EM commence avec quelques sélections initiales pour les paramètres du modèle, qu'on note par θ^{old}. Dans l'étape E (Estimation), on prend les valeurs des paramètres et on trouve la distribution a posteriori des variables latentes $p(\mathbf{Z}^*,\mathbf{Z}/\mathbf{X},\theta^{old})$. Ensuite on utilise cette distribution a posteriori pour évaluer l'espérance du logarithme de la vraisemblance des séquences complètes des données (eq.4.4), en fonction des paramètres θ, pour obtenir la fonction objective $Q(\theta,\theta^{old})$ définie par :

$$Q(\theta,\theta^{old}) = \sum_{\mathbf{Z}^*}\sum_{\mathbf{Z}} p(\mathbf{Z}^*,\mathbf{Z}/\mathbf{X};\theta^{old}) \ln p(\mathbf{X},\mathbf{Z}^*,\mathbf{Z};\theta)$$

$$\begin{aligned} Q(\theta,\theta^{old}) &= \sum_{\mathbf{Z}^*}\sum_{\mathbf{Z}} p(\mathbf{Z}^*,\mathbf{Z}/\mathbf{X};\theta^{old}) \ln p(\mathbf{Z}^*;\pi,\mathbf{A}) \\ &+ \sum_{\mathbf{Z}^*}\sum_{\mathbf{Z}} p(\mathbf{Z}^*,\mathbf{Z}/\mathbf{X};\theta^{old}) \ln p(\mathbf{X}/\mathbf{Z};\phi) \\ &+ \sum_{\mathbf{Z}^*}\sum_{\mathbf{Z}} p(\mathbf{Z}^*,\mathbf{Z}/\mathbf{X};\theta^{old}) \ln p(\mathbf{Z}/\mathbf{Z}^*) \end{aligned}$$

Ainsi on peut réécrire la fonction :

$$Q(\theta,\theta^{old}) = Q_1(\pi,\theta^{old}) + Q_2(\mathbf{A},\theta^{old} + Q_3(\phi,\theta^{old}) + Q_4 \tag{4.5}$$

où

$$Q_1(\pi, \theta^{old}) = \sum_{\mathbf{Z}^*} \sum_{\mathbf{Z}} \sum_{k=1}^{K} p(\mathbf{Z}^*, \mathbf{Z}/\mathbf{X}; \theta^{old}) z_{1k}^* \ln \pi_k$$

$$Q_2(\mathbf{A}, \theta^{old}) = \sum_{\mathbf{Z}^*} \sum_{\mathbf{Z}} \sum_{n=2}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} p(\mathbf{Z}^*, \mathbf{Z}/\mathbf{X}; \theta^{old}) z_{n-1,j}^* z_n^* \ln(A_{jk})$$

$$Q_3(\phi, \theta^{old}) = \sum_{\mathbf{Z}^*} \sum_{\mathbf{Z}} \sum_{n=1}^{N} \sum_{k=1}^{K} p(\mathbf{Z}^*, \mathbf{Z}/\mathbf{X}; \theta^{old}) z_{nk} \ln\left(p(\mathbf{x}_n; \phi_k)\right)$$

$$Q_4 = \sum_{\mathbf{Z}^*} \sum_{\mathbf{Z}} p(\mathbf{Z}^*, \mathbf{Z}/\mathbf{X}; \theta^{old}) \ln p(\mathbf{Z}/\mathbf{Z}^*)$$

A cette étape, on va introduire quelques notations. On va utiliser $\gamma(\mathbf{z}_n^*, \mathbf{z}_n)$ pour noter la distribution marginale *a posteriori* des variables latentes $\mathbf{z}_n^*$ et $\mathbf{z}_n$, et $\xi(\mathbf{z}_{n-1}^*, \mathbf{z}_n^*) = p(\mathbf{z}_{n-1}^*, \mathbf{z}_n^*/\mathbf{X}, \theta^{old})$ pour noter la distribution a posteriori jointe des variables latentes succesives, tel que :

$$\gamma(\mathbf{z}_n^*, \mathbf{z}_n) = p(\mathbf{z}_n^*, \mathbf{z}_n|\mathbf{X}; \theta^{old})$$

ainsi

$$\gamma(\mathbf{z}_n^*) = \sum_{\mathbf{z}} p(\mathbf{z}_n^*, \mathbf{z}_n|\mathbf{X}; \theta^{old})$$

et

$$\gamma(\mathbf{z}_n) = \sum_{\mathbf{z}^*} p(\mathbf{z}_n^*, \mathbf{z}_n|\mathbf{X}; \theta^{old})$$

$$\begin{aligned} \gamma(z_n^{*k}) &= \mathbf{E}[z_n^{*k}] \\ &= \sum_{\mathbf{z}^*} \sum_{\mathbf{z}} \gamma(\mathbf{z}_n^*, \mathbf{z}_n) z_n^{*k} \\ &= \sum_{\mathbf{z}^*} \gamma(\mathbf{z}_n^*) z_n^{*k} \end{aligned}$$

On observe que la fonction objective (eq.4.5) $Q(\theta, \theta^{old})$ est définie comme une somme de quatre termes. Le premier terme $Q_1(\pi, \theta^{old})$ dépend des probabilités initiales ; le deuxième terme $Q_2(\mathbf{A}, \theta^{old})$ dépend des probabilités de transition $\mathbf{A}$; le troisième terme $Q_3(\phi, \theta^{old})$ dépend de ϕ qui est l'ensemble des paramètres de la probabilité d'émission, et le quatrième est une constante. La maximisation de $Q(\theta, \theta^{old})$ par rapport à $\theta = \{\pi, \mathbf{A}, \phi\}$ peut être effectuée séparement.

1) **La maximisation de $Q_1(\pi, \theta^{old})$: Les probabilités initiales**
De la même manière que les modèles probabilistes dédiés aux données i.i.d, on utilise une forme explicite de la distribution des probabilités initiales (chapitre 2, 3). La probabilité initiale π est ensuite obtenue de la manière suivante par rapport au paramètre λ :

$$\pi_k = \frac{e^{\lambda_k}}{\sum_{r=1}^{K} e^{\lambda_r}}$$

$$\begin{aligned} Q_1(\pi, \theta^{old}) &= \sum_{\mathbf{Z}^*} \sum_{\mathbf{Z}} \sum_{k=1}^{K} p(\mathbf{Z}^*, \mathbf{Z}/\mathbf{X}; \theta^{old}) z_{1k}^* \ln \pi_k \\ &= \sum_{\mathbf{Z}^*} \sum_{k=1}^{K} p(\mathbf{Z}^*/\mathbf{X}; \theta^{old}) z_{1k}^* \ln \pi_k \\ &= \sum_{k=1}^{K} \gamma(z_{1k}^*) \ln \pi_k \\ &= \sum_{k=1}^{K} \gamma(z_{1k}^*) \ln \left(\frac{e^{\lambda_k}}{\sum_{r=1}^{K} e^{\lambda_r}} \right) \end{aligned}$$

Si on calcule la dérivée de $Q_1(\pi, \theta^{old})$ par rapport à π, on obtient une mise à jour calculée avec les paramètres précédents. Si nous calculons la dérivée de la fonction $Q_1(\pi, \theta^{old})$ par rapport à λ_k, on peut obtenir après simplification l'expression suivante :

$$\begin{aligned} \frac{\partial Q_1(\pi, \theta^{old})}{\partial \lambda_k} &= \gamma(z_{1k}^*) - \left(\frac{e^{\lambda_k}}{\sum_{r=1}^{K} e^{\lambda_r}} \right) \sum_{j=1}^{K} \gamma(z_{1j}^*) = 0 \\ &= \gamma(z_{1k}^*) - \pi_k \sum_{j=1}^{K} \gamma(z_{1j}^*) = 0 \end{aligned}$$

Ainsi le paramètre de mise à jour est calculé de la manière suivante :

$$\pi_k = \frac{\gamma(z_{1k}^*)}{\sum_{j=1}^{K} \gamma(z_{1j}^*)} \tag{4.6}$$

2) **La maximisation de $Q_2(\mathbf{A}, \theta^{old})$: Probabilités de transition**
Comme dans le cas des HMMs traditionnels, notre modèle utilise un état caché de valeur discrète avec une distribution multinomiale sachant les valeurs précedentes de l'état. Donc notre modèle est un modèle du premier ordre. Ainsi on calcule la

dérivée par rapport à $\mathbf{A}$:

$$\begin{aligned} Q_2(\mathbf{A}, \theta^{old}) &= \sum_{\mathbf{Z}^*} \sum_{\mathbf{Z}} \sum_{n=2}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} p(\mathbf{Z}^*, \mathbf{Z}/\mathbf{X}; \theta^{old}) z^*_{n-1,j} z^*_n \ln(A_{jk}) \\ &= \sum_{\mathbf{Z}^*} \sum_{n=2}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} p(\mathbf{Z}^*/\mathbf{X}; \theta^{old}) z^*_{n-1,j} z^*_n \ln(A_{jk}) \\ &= \sum_{n=2}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} \xi(z^*_{n-1,j} z^*_n) \ln(A_{jk}) \end{aligned}$$

La mise à jour des paramètres est calculée de la manière suivante :

$$A_{jk} = \frac{\sum_{n=2}^{N} \xi(z^*_{n-1,j}, z^{*k}_n)}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z^*_{n-1,j}, z^*_{nl})} \tag{4.7}$$

où

$$\xi(z^*_{n-1,j}, z^*_{n,k}) = \mathbf{E}[z^*_{n-1,j} z^{*k}_n] = \sum_{\mathbf{z}^*} \gamma(\mathbf{z}^*) z^*_{n-1,j} z^*_{n,k}$$

3) **La maximisation de $Q_3(\phi, \theta^{old})$: Les probabilités d'émission**
On sait que :

$$\begin{aligned} Q_3(\phi, \theta^{old}) &= \sum_{\mathbf{Z}^*} \sum_{\mathbf{Z}} \sum_{n=1}^{N} \sum_{k=1}^{K} p(\mathbf{Z}^*, \mathbf{Z}/\mathbf{X}; \theta^{old}) z_{nk} \ln p(\mathbf{x}_n; \phi_k) \\ &= \sum_{\mathbf{Z}} \sum_{n=1}^{N} \sum_{k=1}^{K} p(\mathbf{Z}/\mathbf{X}; \theta^{old}) z_{nk} \ln p(\mathbf{x}_n; \phi_k) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \ln p(\mathbf{x}_n; \phi_k) \end{aligned}$$

L'ensemble des paramètres ϕ dépend de la distribution utilisée. Nous présentons ci-dessous deux applications en utilisant la loi Gaussienne et la loi de Bernoulli.

– **Distribution continue**
Dans le cas des probabilités d'émission avec une densité sphérique gaussienne on a $p(\mathbf{x}/\phi_k) = \mathcal{N}(\mathbf{x}; \mathbf{w}_k, \sigma_k)$, définie par sa "moyenne" $\mathbf{w}_k$, qui a la même dimension que les données d'entrée, et sa matrice de covariance, définie par $\sigma_k^2 \mathbf{I}$ où σ_k est l'écart-type et $\mathbf{I}$ est la matrice identité,

$$N(\mathbf{x}; \mathbf{w}_k, \sigma_k) = \frac{1}{(2\pi\sigma_k)^{\frac{d}{2}}} \exp\left[\frac{||\mathbf{x} - \mathbf{w}_k||^2}{2\sigma_k^2}\right]$$

La maximisation de la fonction $Q_3(\phi, \theta^{old})$ fournit les expressions connues :

$$\mathbf{w}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{4.8}$$

$$\sigma_k^2 = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) ||\mathbf{x}_n - \mathbf{w}_k||^2}{d \sum_{n=1}^{N} \gamma(z_{nk})} \tag{4.9}$$

où d est la dimension de l'élément $\mathbf{x}$.

– **Données binaires : Distribution de Bernoulli**
Nous supposons que chaque observation $\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^d)$ est un vecteur binaire $\{0,1\}^d$. Dans le cas des données binaires on propose d'utiliser la loi de Bernoulli décrite dans le chapitre 2 qui est utilisée pour définir le modèle probabiliste BeSOM-ε_k : $p(\mathbf{x}/\phi_k) = f_k(\mathbf{x}, \mathbf{w}_k, \varepsilon_k)$ qui admet les paramètres $\mathbf{w}_k = (w_k^1, w_k^2, ..., w_k^d) \in \{0,1\}^d$ avec la probabilité $\varepsilon_k \in]0, \frac{1}{2}[$ associée à $\mathbf{w}_k$. Le paramètre ε_k définit la probabilité d'être différent du prototype $\mathbf{w}_k$. La distribution associée à chaque état k de la chaîne de Markov $\mathcal{C}$ est définie de la façon suivante :

$$f_k(\mathbf{x}, \mathbf{w}_k, \varepsilon_k) = \varepsilon_k^{\mathcal{H}(\mathbf{x}, \mathbf{w}_k)} (1 - \varepsilon_k)^{d - \mathcal{H}(\mathbf{x}, \mathbf{w}_k)} \tag{4.10}$$

Le symbole $\mathcal{H}$ indique la distance de Hamming qui permet la comparaison des vecteurs binaires $\mathbf{x}_n$ et $\mathbf{x}_m$. La distance de Hamming calcule le nombre de désaccords entre $\mathbf{x}_n$ et $\mathbf{x}_m$:

$$\mathcal{H}(\mathbf{x}_n, \mathbf{x}_m) = \sum_{l=1}^{d} |x_n^l - x_m^l|$$

Ainsi la distance de Hamming nous permet de calculer le centre médian binaire, et dans ce cas, la plus importante caractéristique de centre médian est un vecteur binaire qui a la même interprétation (le même codage) que les observations dans l'espace d'entrée. Utilisant la loi de Bernoulli (eq.4.10), la fonction objective $Q_3(\phi, \theta^{old})$ devient :

$$\begin{aligned} Q_3(\phi, \theta^{old}) &= \sum_{k=1}^{K} \sum_{n=1}^{N} \left[-\gamma(z_{nk}) \ln \left(\frac{1 - \varepsilon_k}{\varepsilon_k} \right) \mathcal{H}(\mathbf{x}_n - \mathbf{w}_k) \right] \\ &+ \sum_{k=1}^{K} \sum_{n=1}^{N} \left[d\gamma(z_{nk}) \ln(1 - \varepsilon_k) \right] \end{aligned}$$

La maximisation de $Q_3(\phi, \theta^{old})$ par rapport à w_{kj} et ε_k est effectuée séparément. On constate que avec la condition $\varepsilon_k \in]0, \frac{1}{2}[$, le premier terme de $Q_3(\phi, \theta^{old})$ est

un terme négatif, et par conséquent la maximisation de $Q_3(\phi, \theta^{old})$ nécessite la minimisation de l'inertie $\mathcal{I}_k^j$ associée à chaque variable binaire x^j définie de la manière suivante :

$$\mathcal{I}_k^j(w_k^j) = \sum_{n=1}^{N} \gamma(z_{nk}) |x_n^j - w_k^j|$$

sous la contrainte $w_k^j \in \{0, 1\}$, il est clair que cette minimisation nous permet de définir le centre médian $\mathbf{w}_k \in \{0, 1\}^n$ des observations $\mathbf{x}_i$ pondérées par $\gamma(z_{nk})$ [125]. Chaque composante de $\mathbf{w}_k = (w_k^1, w_k^2, ..., w_k^j, ..., w_k^d)$ est ensuite calculée d'après cette expression :

$$w_k^j = \begin{cases} 0 & \text{si } \left[\sum_{n=1}^{N} \gamma(z_{nk})(1 - x_n^j)\right] \geq \\ & \left[\sum_{n=1}^{N} \gamma(z_{nk}) x_n^j\right] \\ 1 & \text{sinon} \end{cases} \tag{4.11}$$

En deuxième étape, nous calculons la dérivée de la fonction $Q_3(\phi, \theta^{old})$ par rapport à ε_k $\left(\frac{\partial Q_3(\phi, \theta^{old})}{\partial \varepsilon_k} = 0\right)$. Cela mène à une mise à jour qui est calculée avec les paramètres estimés à l'étape $(t-1)$ comme défini ci-dessous :

$$\varepsilon_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathcal{H}(\mathbf{x}_n, \mathbf{w}_k)}{d \sum_{n=1}^{N} \gamma(z_{nk})} \tag{4.12}$$

L'algorithme EM demande des valeurs initiales pour la distribution de l'émission. Une manière de calculer cela est : de traiter les données sous l'hypothèse i.i.d (chapitre 2 et 3) et d'adapter la densité d'émission par le maximum de la vraisemblance et ensuite utiliser les valeurs résultantes pour initialiser les paramètres de notre modèle.

4.2.1.2 L'algorithme forward-backward : l'étape E (Estimation)

On cherche une procédure efficace pour évaluer les quantités $\gamma(\mathbf{z}_n^*)$, $\gamma(\mathbf{z}_n)$ et $\xi(\mathbf{z}_{n-1}^*, \mathbf{z}_n^*)$, qui correspondent à l'étape E (Estimation) de l'algorithme EM. Dans le contexte particulier du modèle de Markov caché, on va utiliser l'algorithme forward-backward [134], ou encore appelé l'algorithme Baum-Welch [14, 2]. Dans notre cas, il peut être renommé par l'algorithme "forward-backward" topologique, puisqu'on utilise la structure du graphe pour organiser les données séquentielles d'une manière explicite. Quelques formules sont similaires aux chaînes de Markov traditionnelles si on n'utilise pas la structure du graphe. On va utiliser les notations $\alpha(z_n^{*k})$ et $\alpha(z_{nk})$ pour noter la valeur

de $\alpha(\mathbf{z}^*)$ et $\alpha(\mathbf{z})$ quand $z_n^{*k} = 1$, $z_{nk} = 1$ avec des notations analogues pour β.

$$\begin{aligned}\gamma(\mathbf{z}_n^*) &= p(\mathbf{z}_n^*/\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{z}_n^*)p(\mathbf{z}_n^*)}{p(\mathbf{X})} \\ &= \frac{p(\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{z}_n^*)p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N/\mathbf{z}_n^*)}{p(\mathbf{X})} \\ \gamma(\mathbf{z}_n^*) &= \frac{\alpha(\mathbf{z}_n^*)\beta(\mathbf{z}_n^*)}{p(\mathbf{X})}\end{aligned}$$

Utilisant une décomposition similaire on obtient

$$\gamma(\mathbf{z}_n) = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}$$

Les valeurs de $\alpha(\mathbf{z}_n^*)$ et $\alpha(\mathbf{z}_n)$ sont calculées par récursion " **forward**" de la manière suivante :

$$\begin{aligned}\alpha(\mathbf{z}_n^*) &= \left[\sum_{\mathbf{z}} p(\mathbf{x}_n/\mathbf{z}_n)p(\mathbf{z}_n/\mathbf{z}_n^*)\right] \\ &\times \sum_{\mathbf{z}_{n-1}^*} \alpha(\mathbf{z}_{n-1}^*)p(\mathbf{z}_n^*|\mathbf{z}_{n-1}^*)\end{aligned} \tag{4.13}$$

et

$$\begin{aligned}\alpha(\mathbf{z}_n) &= p(\mathbf{x}_n|\mathbf{z}_n)\sum_{\mathbf{z}_n^*} p(\mathbf{z}_n/\mathbf{z}_n^*) \\ &\left[\sum_{\mathbf{z}_{n-1}^*} \alpha(\mathbf{z}_{n-1}^*)p(\mathbf{z}_n^*|\mathbf{z}_{n-1}^*)\sum_{\mathbf{z}_{n-1}} p(\mathbf{z}_{n-1}|\mathbf{z}_{n-1}^*)\right]\end{aligned} \tag{4.14}$$

où

$$p(\mathbf{z}_n/\mathbf{z}_n^*) = p(z_{nc} = 1/z_{nc^*}^* = 1) = \frac{\mathcal{K}^T(\delta(c, c^*))}{\sum_{r\in\mathcal{C}} \mathcal{K}^T(\delta(r, c^*))}$$

et $\sum_{\mathbf{z}_{n-1}} p(\mathbf{z}_{n-1}|\mathbf{z}_{n-1}^*) = 1$.
Pour lancer cette recursion, on a besoin d'une contrainte initiale qui est donnée par

$$\alpha(\mathbf{z}_1^*) = p(\mathbf{x}_1, \mathbf{z}_1^*) = p(\mathbf{z}_1^*)\left[\sum_{\mathbf{z}_1} p(\mathbf{x}_1/\mathbf{z}_1)p(\mathbf{z}_1/\mathbf{z}_1^*)\right]$$

$$\alpha(\mathbf{z}_1) = p(\mathbf{x}_1, \mathbf{z}_1) = p(\mathbf{x}_1/\mathbf{z}_1)\left[\sum_{\mathbf{z}_1^*} p(\mathbf{z}_1^*)p(\mathbf{z}_1/\mathbf{z}_1^*)\right]$$

La valeur de $\beta(\mathbf{z}_n^*)$, est calculée par la récursion " **backward**" de la façon suivante :

$$\beta(\mathbf{z}_n^*) = \sum_{\mathbf{z}_{n+1}^*} \beta(\mathbf{z}_{n+1}^*) p(\mathbf{x}_{n+1}/\mathbf{z}_{n+1}^*) p(\mathbf{z}_{n+1}^*/\mathbf{z}_n^*) \tag{4.15}$$

$$\begin{aligned} \beta(\mathbf{z}_n) = & \frac{1}{p(\mathbf{z}_n)} \sum_{\mathbf{z}_n^*} p(\mathbf{z}_n^*) p(\mathbf{z}_n/\mathbf{z}_n^*) \sum_{\mathbf{z}_{n+1}} \sum_{\mathbf{z}_{n+1}^*} \\ & p(\mathbf{z}_{n+1}/\mathbf{z}_{n+1}^*) \beta(\mathbf{z}_{n+1}^*) p(\mathbf{x}_{n+1}/\mathbf{z}_{n+1}^*) p(\mathbf{z}_{n+1}^*/\mathbf{z}_n^*) \end{aligned} \tag{4.16}$$

où

$$p(\mathbf{x}_{n+1}/\mathbf{z}_{n+1}^*) = \left[\sum_{\mathbf{z}} p(\mathbf{x}_{n+1}/\mathbf{z}_{n+1}) p(\mathbf{z}_{n+1}/\mathbf{z}_{n+1}^*) \right]$$

$$p(\mathbf{z}_n) = \sum_{\mathbf{z}_n^*} p(\mathbf{z}_n^*) p(\mathbf{z}_n/\mathbf{z}_n^*)$$

et

$$\begin{aligned} p(\mathbf{z}_{n+1}/\mathbf{z}_{n+1}^*) &= p(z_{n+1,c} = 1/z_{n+1,c^*}^* = 1) \\ &= \frac{K^T(\delta(c,c^*))}{\sum_{r \in \mathcal{C}} K^T(\delta(r,c^*))} \end{aligned}$$

Il est évident que la fonction de voisinage ne dépend pas du temps $p(\mathbf{z}_{n+1}/\mathbf{z}_{n+1}^*) = p(\mathbf{z}_n/\mathbf{z}_n^*)$. On a encore besoin d'une condition de départ pour la récursion, on prend une valeur pour $\beta(\mathbf{z}_N^*) = 1$ et $\beta(\mathbf{z}_N) = 1$. Cela peut être obtenue en fixant $n = N$ dans l'expression (4.13).

Ensuite on envisage l'évaluation des quantités $\xi(\mathbf{z}_{n-1}^*, \mathbf{z}_n^*)$ qui correspondent aux valeurs des probabilités conditionnelles $p(\mathbf{z}_{n-1}^*, \mathbf{z}_n^*/\mathbf{X})$ pour chaque instance $(\mathbf{z}_{n-1}^*, \mathbf{z}_n^*)$. Ainsi on applique le théorème de Bayes, et on obtient

$$\begin{aligned} \xi(\mathbf{z}_{n-1}^*, \mathbf{z}_n^*) &= p(\mathbf{z}_{n-1}^*, \mathbf{z}_n^*/\mathbf{X}) \\ &= \frac{p(\mathbf{X}/\mathbf{z}_{n-1}^*, \mathbf{z}_n^*) p(\mathbf{z}_{n-1}^*, \mathbf{z}_n^*)}{p(\mathbf{X})} \\ \xi(\mathbf{z}_{n-1}^*, \mathbf{z}_n^*) &= \frac{\alpha(\mathbf{z}_{n-1}^*) \left[\sum_{\mathbf{z}} p(\mathbf{x}_n/\mathbf{z}_n) p(\mathbf{z}_n/\mathbf{z}_n^*)\right]}{p(\mathbf{X})} \\ &\times \frac{p(\mathbf{z}_n^*/\mathbf{z}_{n-1}^*) \beta(\mathbf{z}_n^*)}{p(\mathbf{X})} \end{aligned}$$

Si on additionne $\alpha(\mathbf{z}^*)$ par rapport à $\mathbf{z}_N$, on obtient $p(\mathbf{X}) = \sum_{\mathbf{z}_N} \alpha(\mathbf{z}_N)$. Après le calcul de la récursion "forward" α et la récursion backward β on utilise les résultats pour évaluer la probabilité a posteriori γ et $\xi(\mathbf{z}^*_{n-1}, \mathbf{z}^*_n)$. On utilise ces résultats pour calculer un autre paramètre du modèle θ utilisant les équations de l'étape M (Maximisation) (4.6, 4.7, 4.8, 4.9). Ces deux étapes sont répétées jusqu'à ce qu'un critère de convergence soit satisfait.

4.2.2 Analyse de l'auto-organisation du modèle de Markov proposé

L'approche globale que nous proposons nous permet d'estimer les paramètres qui maximisent la fonction log-vraisemblance pour un paramètre T fixe. Comme dans le cas de l'algorithme de classification topologique probabiliste sous l'hypothèse de données i.i.d, on doit varier la valeur du paramètre T entre deux valeurs T_{max} et T_{min}, pour contrôler l'influence du voisinage d'un état donné dans le graphe au même instant. Pour chaque valeur de T, on obtient une fonction de vraisemblance Q^T, et par conséquent l'expression varie avec T. Deux étapes sont définies (figure 4.4) :

- La première phase correspond à des valeurs élevées de T. Dans ce cas, l'influence du voisinage de chaque état $\mathbf{z}^*$ dans le graphe HMM, associé à notre approche, est importante et correspond à des valeurs plus élevées de la fonction $\mathcal{K}^T(\delta(c,r))$. Ainsi, les formules décrites pour l'estimation des paramètres utilisent un grand nombre d'états au même instant et par conséquent utilisent un grand nombre d'observations pour estimer les paramètres du modèle. Cette phase permet d'obtenir l'ordre topologique du modèle de Markov.

- La deuxième étape correspond à des petites valeurs de T. A chaque instant le nombre d'états est limité, donc, l'adaptation est très locale. Les paramètres sont calculés avec précision à partir de la densité des données. Dans ce cas on peut considérer qu'on converge vers des HMMs traditionnels, puisque le voisinage est presque null. On rappelle que la classification à base de modèle de mélange BeSOM ou PrSOM pour des données i.i.d. est un cas particulier des HMM [23, chap 9].

4.3 Approche locale

Nous proposons ici une approche hybride faisant coopérer des cartes auto organisatrices et des chaînes de Markov cachées (HMM, Hidden Markov Models). Elle permet

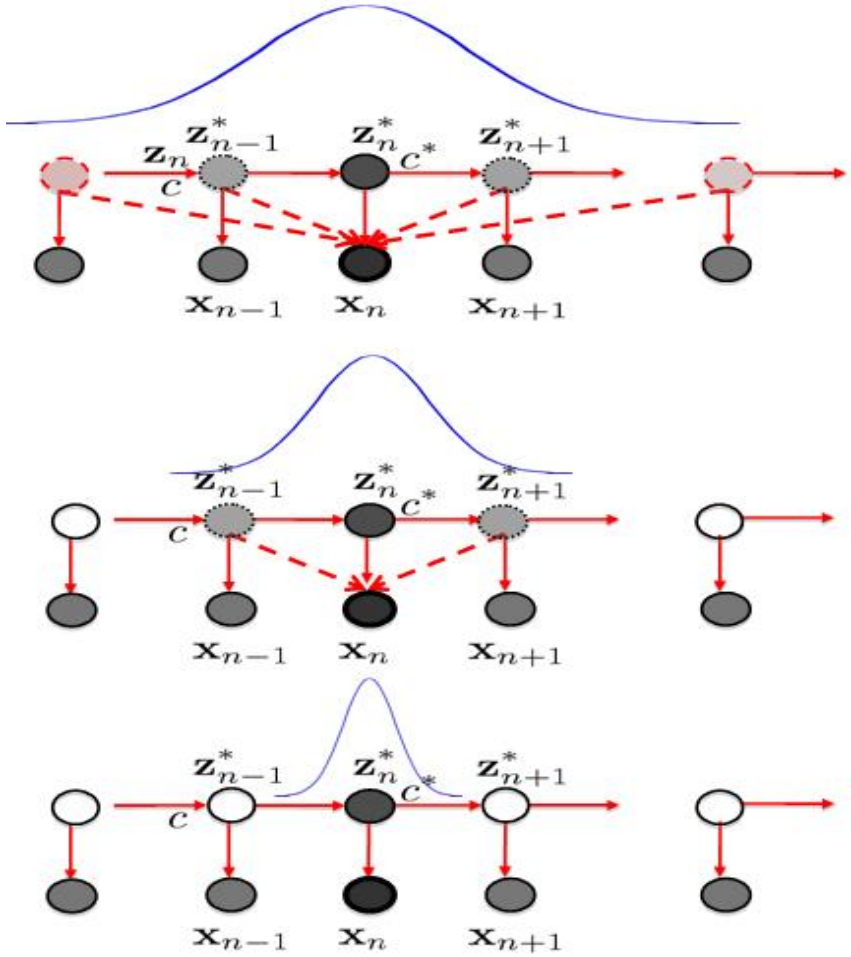


FIG. 4.4 – Une forme simplifiée du graphe associé à notre approche globale pour décrire l'influence du voisinage. La fonction sous la forme "Gaussienne" montre la variation de la fonction de voisinage $\mathcal{K}$ centrée sur un état

de traiter des données structurées en séquences en alliant les points forts des deux méthodes (figure 4.2(a)). Cette approche permet d'obtenir : d'une part, la découverte par apprentissage de la structure des modèles HMM dans un modèle de mélanges adapté aux données ; et d'autre part l'optimisation de la structure de la carte par élagage des états.

4.3.1 Description de l'approche locale

Le principe de l'approche locale est d'utiliser une carte topologique probabiliste en amont pour l'initialisation des HMMs. L'information topologique, fournie par les cartes

topologiques probabilistes, permettra d'élaborer une topologie pour les modèles HMM où chaque HMM est modélisé par une région précise ou un cluster de la carte. A la différence avec l'approche globale, ce sont en effet les cartes topologiques qui permettent d'introduire d'une manière implicite la notion de voisinage entre les différentes HMMs associé à chaque cluster. Afin de modéliser la carte topologique sous forme de plusieurs chaînes de Markov cachées, on adopte la même hypothèse prise dans l'approche globale qui consiste à définir le graphe représentant la chaîne de Markov par la grille de cellules C, telle que chaque cellule c représente un état de la chaîne. Ce modèle est décrit dans la section 4.2.1. Ainsi toutes les probabilités permettant de définir le HMM sont des résultats extraits de la carte topologique probabiliste.

L'idée principale dans la classification à base de modèle de mélange est que les données séquentielles observées sont supposées générées à l'aide d'un mélange de K "prototypes" distributions statistiques, chacun représente un cluster, et à l'intérieur duquel une certaine variabilité est possible [34]. Ce modèle est un modèle de mélange à K composantes. Ainsi chaque cluster sera modélisé par une chaîne de Markov cachée. Au départ, biensûr, on ne connait pas le modèle ; seulement les données sont disponibles, mais c'est possible d'appliquer des techniques statistiques standard pour "apprendre" les paramètres du modèle : $\theta = (\mathbf{A}, \phi, \pi)$ où $\mathbf{A}$ est la matrice de transitions, ϕ est l'ensemble des paramètres de la distribution qui est connue comme la probabilité d'émission et π est la matrice des probabilités initiales. Cette modélisation est un cas particulier de l'approche globale présentée dans la section 4.2.1. On peut reécrire la distribution marginale comme :

$$p(\mathbf{X}; \theta) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{X}/c_k; \theta_k) \tag{4.17}$$

où $\pi_k = p(c_k)$ est la probabilité a priori du k^{ime} cluster, ou le "poids" de cette composante dans le modèle ($\sum_k \pi_k = 1$) ; la probabilité $p_k(\mathbf{X}/c_k)$ est la distribution de X dans le k^{ieme} cluster ; où θ_k sont les paramètres de la distribution associée au k^{ime} cluster. Quand les observations sont des séquences d'événements (ici des cellules), un candidat naturel pour p_k est une chaîne de Markov, dont les états correspondent aux cellules de la carte topologique. Finalement, on doit estimer les paramètres θ_k associés à chaque cluster modélisé par une chaîne de Markov cachée. De la même manière que l'approche globale, on utilise l'algorithme EM pour trouver les paramètres qui maximisent la fonction de vraisemblance. L'algorithme EM commence avec quelques sélections initiales pour les paramètres du modèle, qu'on note par θ^{old}.

La validation de l'approche locale proposée a été réalisée sur les phrases de récits

d'accident de la route fournis par la MAIF. Les résultats sont prometteurs, à la fois pour la classification et pour la modélisation.

4.3.2 Exemple d'application : Base d'assurance MAIF

Les récits qui ont été analysés sont des récits d'accident de la route provenant de constats destinés aux assureurs. Ils nous ont été gracieusement fournis par la MAIF que nous remercions. On a considéré dans ces récits les séquences de verbes délimitées par les fins de phrases.

L'analyse a été réalisée sur une centaine de textes correspondant à 700 occurrences de verbes réparties sur 260 phrases. Pour le codage, les quatre catégories aspectuelles de verbes (état, activité, accomplissement, achèvement) - dues originellement à Vendler et Kenny [158] ont été couplées avec le temps grammatical du verbe. L'indexation a été réalisée à la main sur la base de notre conception de ces catégories, telle qu'explicitée dans [56]. On a cherché à voir si l'on pouvait détecter une certaine régularité dans les suites d'emplois croisés temps et catégorie aspectuelle des verbes, pour l'expression générale d'un incident et de ces causes, le récit étant a priori contraint par un espace limité. De nombreuses études ont montré l'importance des temps grammaticaux dans la structure narrative d'un récit. L'idée d'un couplage des temps et des catégories aspectuelles est naturelle, car de nombreux liens unissent le temps et l'aspect. Il a d'ailleurs été démontré qu'on ne peut effectuer l'analyse des successions d'événements à partir du seul indice du temps grammatical sans faire intervenir l'aspect. Les techniques d'apprentissage numérique utilisées ont permis de classifier les transitions entre deux verbes successifs (verbe1, verbe2) en analysant les séquences (de longueur variable) constituées par les verbes des phrases à l'aide de modèles de Markov cachés. Les profils des transitions obtenues sont représentés sur des cartes topologiques incorporant une notion de voisinage. La matrice des distances entre les profils de transitions a fourni une répartition de ces transitions en 4 groupes (clusters) distincts. Le nombre de 4 est apparu ici comme le meilleur selon un critère de qualité de la classification non supervisée [50], et il est intéressant pour l'interprétation qu'il soit relativement petit (nous étions en effet parti de 9 temps et 4 catégories).

4.3.3 Validation expérimentale de l'approche locale

Les cartes probabilistes nous permettent de définir les éléments initiaux des HMM. En effet, la formulation probabiliste nous permet de représenter la carte sous forme d'un modèle de mélange où chaque cluster est représenté par un HMM (figure 4.5). Cette modélisation des cartes sous forme de modèle de mélange nous donne une première initialisation des HMM qui sera ensuite optimisée par apprentissage. Les résultats de cette optimisation sont présentés dans la figure 4.5.

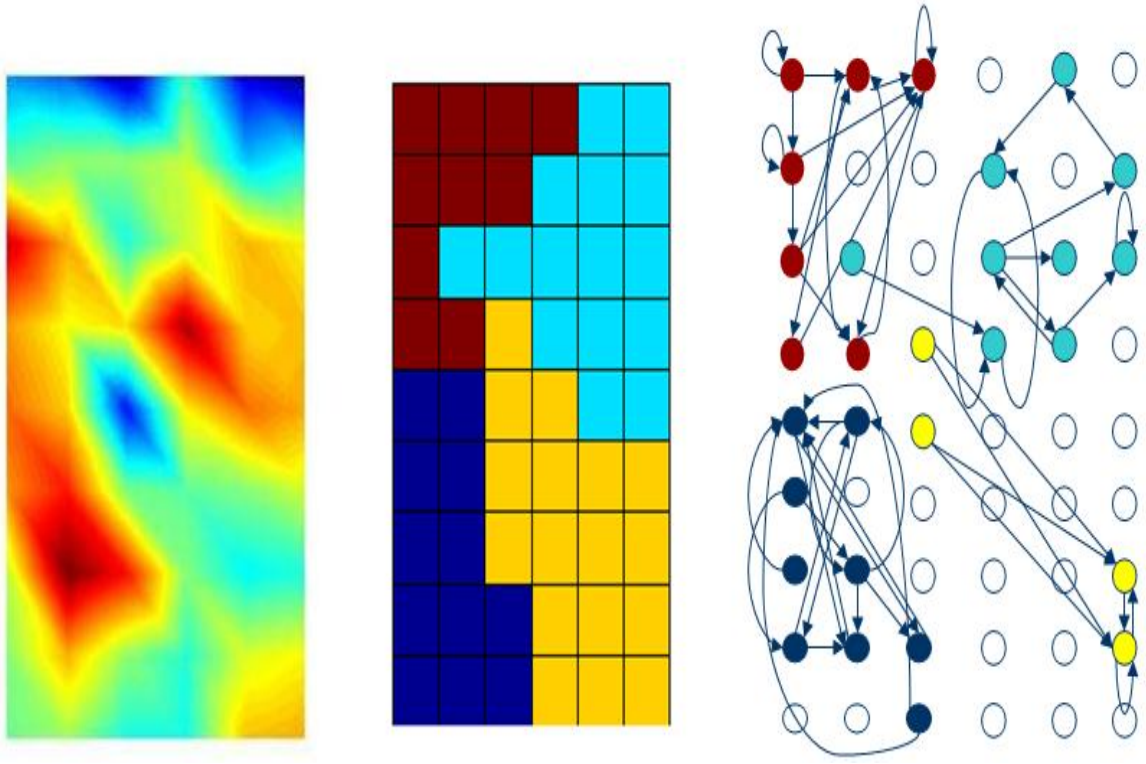


FIG. 4.5 – A gauche on a la matrice des distances entre les prototypes (plus c'est rouge plus les prototypes son éloignés), au milieu on a le découpage de la carte en 4 clusters et à droite on a les clusters après l'optimisation des clusters (on observe qu'après l'optimisation il existe certaines cellules qui sont élaguées), chaque cluster représente un HMM

A partir des clusters obtenus par l'apprentissage des cartes probabilistes, on extrait la topologie d'un modèle de Markov en se basant sur les probabilités d'émissions et de transitions des cellules. Les matrices de transitions et d'émission nous permettent de construire le modèle de Markov caché associé. Notons par ailleurs que lors du processus d'apprentissage, certaines cellules sont élaguées (les probabilités de transitions

s'annulent).

Cette approche nous permet d'une part de découvrir une topologie pour les HMM et d'autre part d'optimiser la carte auto-organisatrice. Chaque cluster modélise une chaîne de Markov. En effet, à partir des paramètres d'initialisation (matrices des probabilités de transitions et d'émissions) extraits de la carte probabiliste, un apprentissage classique des HMM est effectué pour optimiser les modèles et par conséquent introduire un mécanisme d'élagage sur les cellules de la carte probabiliste. L'approche locale apporte ici des informations plus précises sur la dynamique interne des séquences à chaque cluster (figure 4.6).

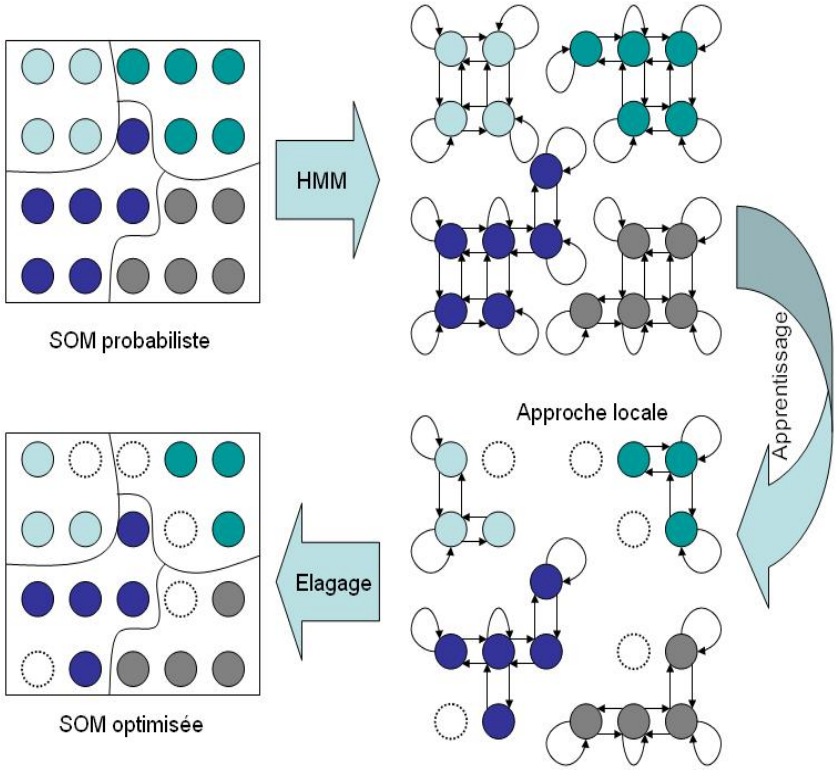


FIG. 4.6 – Processus apprentissage des HMMs et d'élagage des états.

Nous allons présenter maintenant une interprétation de ces quatre groupes de transitions verbales, réalisées en grande partie sur la base d'information statistique sur des marques sémantiques indépendantes. Les informations concernant la topologie (transférée par celle des cartes aux chaînes de Markov) nous ont permis par ailleurs de trouver des représentants typiques pour ces quatre groupes de transitions (verbe1, verbe2).

4.3.4 Analyse des résultats[1]

4.3.4.1 Statistiques globales et éléments pour l'interprétation

Ce type de récit n'utilise globalement que l'imparfait (24%) et le passé composé (34%), avec de temps à autre quelques phrases au présent. On y trouve aussi quelques occurrences (rares) de passé simple et de plus-que-parfait. Il existe par contre un nombre important de participes présents (11%) et d'infinitifs (20%). Nous avons donc décidé de les retenir, bien qu'ils ne participent pas de la même manière à l'ossature grammaticale. Nous avons finalement effectué l'apprentissage en gardant 9 codes apparus sur les verbes : IM = imparfait, PR = présent, PC = passé composé, PS = passé simple, PQP = plus-que-parfait, inf = infinitif, ppr = participe présent, pp = participe passé et pps = participe passé surcomposé. . Dans ce type de récit, les verbes d'état représentent 24% du corpus, ceux d'activité seulement 10%, et l'on trouve 34% de verbes d'accomplissement et 32% de verbes d'achèvement. Les pourcentages des (principaux) temps par catégorie sont donnés graphiquement Figure 4.7 et le tableau 4.1 résume les différences sémantiques entre les quatre catégories aspectuelles.

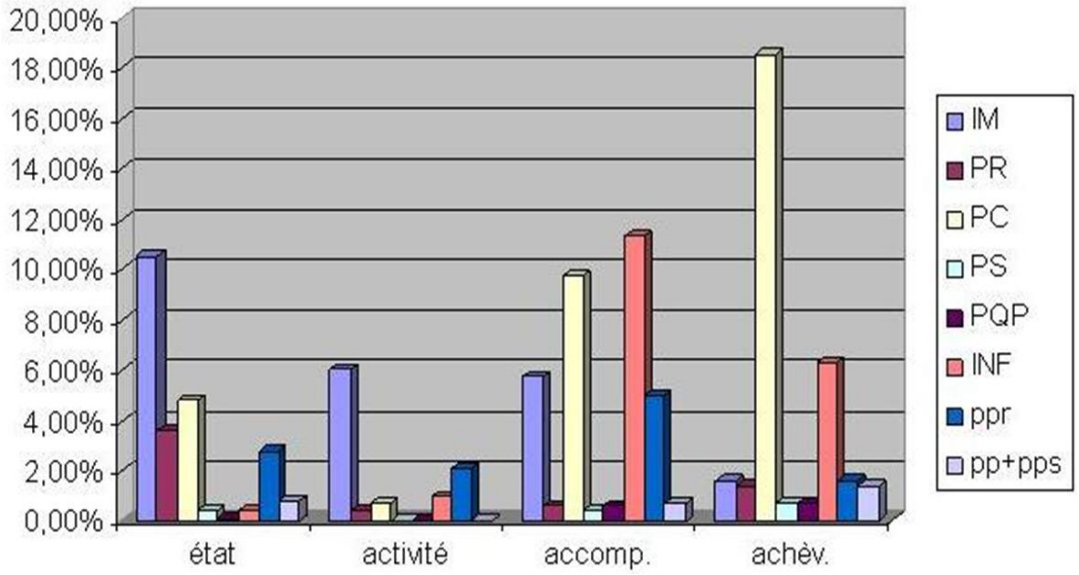


FIG. 4.7 – Proportion des couples (catégorie, temps)

Les verbes d'états (24%). Ils sont répartis à plus de 70% sur l'imparfait, le présent

[1]Cette analyse des résultats a été effectuée en collaboration avec Mme Chatherine RECANATI MCF à l'université Paris 13 et membre de l'équipe RCLN (Représentation des Connaissances et Langage Naturel)

et le participe présent. Cela n'est guère surprenant puisque les états sont homogènes, souvent duratifs ou caractérisant une aptitude (habituels, génériques). La proportion non négligeable du passé composé s'explique en partie par la fréquence de verbes comme "vouloir" ou "pouvoir" que nous avons classés comme verbes d'états ("j'ai voulu freiner", "je n'ai pu éviter"). La faible proportion de présent provient du fait que le récit est au passé et que le présent historique est trop littéraire pour le genre.

Les verbes d'activités (10%). De la même façon, les activités dénotant des processus homogènes au plan macroscopique et non bornés (à droite), elles se répartissent tout naturellement à plus de 79% sur l'imparfait et le participe présent. La présence de 10% d'infinitif peut facilement s'expliquer par le fait qu'il s'agit de processus qui ont un début et qui peuvent donc se trouver complément de verbe comme "commencer à", "vouloir", ou simplement être introduit pour mentionner un but avec la préposition "pour".

Accomplissement (34%) et achèvements (32%). A l'inverse des deux catégories précédentes, le caractère télique de ces verbes explique leur fréquence au passé composé. Les achèvements y sont présents de manière massive car étant ponctuels (ou de courte durée), ils indiquent principalement un changement d'état et supportent mal l'imparfait. A l'inverse, les accomplissements supportent bien l'imparfait et le participe présent, parce qu'ils ont une durée intrinsèque, et mettent l'accent sur le procès plutôt que sur sa fin - ce qui les rapproche finalement des activités. L'importance globale de ces deux catégories est certainement liée à la typologie des textes, un récit d'accident impliquant de décrire la séquence des événements successifs l'ayant provoqué.

Structure prototypique. Un récit d'accident commence généralement par quelques phrases décrivant les circonstances et les états de choses précédant l'accident. Cette première partie est alors à l'imparfait, et contient de nombreux participes présents. On y trouve aussi quelques présents et de nombreux infinitifs introduits par "pour", ou compléments de verbes ("je m'apprêtais à tourner", "le feu venait de passer au rouge"). Cette première partie, essentiellement circonstancielle, contient une majorité de verbes d'états, quelques activités et quelques accomplissements. Elle est globalement caractérisée par un point de vue imperfectif, et un relief en arrière-plan. Vient ensuite la description de l'accident proprement dit qui mentionne la succession des événements ayant conduit à l'accident, et termine par le moment du choc. Cette partie utilise massivement des verbes d'accomplissement et d'achèvement, le plus souvent au passé composé. Elle se trouve globalement caractérisée par un mode perfectif, mais le but du jeu étant d'indiquer les responsabilités des différents acteurs, on trouve ici encore beaucoup de participes présents et de tournures infinitives enchaînant parfois plusieurs verbes ("J'ai voulu freiner pour l'éviter", "voulant éviter la borne, je n'ai pu"). En fin

verbe d'ETAT procès homogène, duratif, habituel ou dispositionnel	**verbe d'ACTIVITE** processus homogène, non borné
être(à l'arrêt)/ vouloir/ pouvoir	rouler/ circuler/ slalomer/ suivre
verbe d'ACCOMPLISSEMENT processus borné par une fin	**verbe d'ACHEVEMENT** événement quasi ponctuel
traverser/ faire un créneau/ aller à	franchir / heurter / percuter

TAB. 4.1 – Les quatre catégories aspectuelles de verbe

de récit, on trouve enfin parfois une dernière partie constituée de commentaires sur la situation et inventoriant notamment les dégâts. Cette partie est souvent courte et moins facile à caractériser. Pour faciliter l'interprétation des quatre groupes de transitions obtenus, nous avons indexé tous les verbes par un numéro indiquant la partie concernée (1 - circonstance, 2 - accident ou 3 - commentaire).

Autres marques sémantiques. Outre ce découpage et les indications d'avant-plan et d'arrière-plan, nous avons indexé les verbes d'un certain nombre d'attributs. Pour déceler d'éventuelles chaînes causales conduisant à l'accident nous avons marqué des verbes avec les attributs causal ou choc. Nous avons également marqué les verbes d'action en fonction de l'agent responsable (A pour l'auteur du récit, B pour l'adversaire, et C pour un tiers). Nous avons aussi noté la présence de négation, de buts, et l'évocation plus générale de mondes possibles voisins qui ne se sont pas produits (attributs négation et inertia). Ces marques n'ont pas été utilisées pour l'apprentissage, mais elles nous permettent de valider les groupes de transitions verbales s'ils se différencient ensuite aussi sémantiquement. Nous allons maintenant présenter les quatre groupes (ou clusters) de transitions obtenus.

1) **Cluster (C) des Circonstances**
Description. Le groupe des transitions (C) se distingue par une différenciation nette du premier verbe et du second. Le premier verbe est à 93% à l'imparfait, pour seulement 7% de présent, tandis que le second est à 63% à l'infinitif et à 30% au participe présent. Du point de vue des catégories aspectuelles, le premier verbe est à 56% un verbe d'état, et le second à 63% un verbe d'accomplissement. (Les autres catégories se trouvant distribuées de manière régulière entre 12% et 16%). On peut donc résumer globalement les transitions de ce groupe comme présentant un verbe d'état (ou d'activité) à l'imparfait, suivi d'un verbe d'accomplissement à l'infinitif ou au participe présent. L'analyse des prototypes nous fournit une synthèse plus fine (Tableau 4.2). Ce groupe privilégie les états et les

activités au détriment des achèvements - les accomplissements étant par contre massivement représentés en seconde occurrence verbale.

Interprétation. Le groupe (C) est celui où l'attribut inertia (marquant un but ou un monde possible voisin) est le plus important. Cela s'explique par les nombreux accomplissements introduits par la préposition "pour" ("je reculais pour repartir", "je sortais du parking pour me diriger") ou les auxiliaires à l'imparfait introduisant un infinitif et indiquant des intentions du conducteur ("je m'apprêtais à tourner à gauche"). C'est une des raisons pour laquelle nous l'avons baptisé groupe (C) des Circonstances. L'autre raison est que ce groupe contient une forte majorité de verbes appartenant à la première partie du récit (partie 1 circonstance), et très peu de la seconde (2 accident) ou de la troisième (3 commentaire). On constate en outre que ce groupe ne contient pratiquement aucun verbe indiquant les causes de l'accident ou le choc. L'acteur A y est le plus représenté, et le récit se trouve en arrière-plan.

Synthèse	**verbe 1**	**verbe 2**
type 1	état ou act., IM	état ou act., ppr
type 2	état, IM (ou PR)	acc., INF
type 3	act. ou ach., IM	acc.(ou ach) INF

TAB. 4.2 – Synthèse des prototypes du cluster (C)

2) **Cluster (CI) des Circonstances ou de la survenue d'un Incident**
Description. Le deuxième groupe (CI) est celui des circonstances ou de la survenue d'un incident. On note ici un grand nombre de verbes d'états (37,5%) et d'activités (17%), encore plus important que dans le groupe précédent et très supérieur à la moyenne. On a par contre un nombre moyen d'achèvements (29%), absents du premier verbe mais massivement représentés sur le second. Cela distingue ce groupe du groupe (C) précédent, où les accomplissements jouaient ce rôle. Ici, à l'inverse, les accomplissements se trouvent exclus de la seconde place, et sont nettement sous-représentés (16,5%). On peut synthétiser les transitions de ce groupe en disant qu'on a généralement affaire à un état ou une activité à l'imparfait. On enchaîne donc deux points de vue imperfectifs (ou neutre), et parfois, un point de vue imperfectif et un point de vue perfectif.

Interprétation. On a dans ce groupe de transitions aussi bien un grand nombre de séquences imperfectives provenant de la partie 1 circonstance ("Je circulais à environ 45 km/h dans une petite rue à sens unique où stationnaient des voitures de chaque côté"), que de séquences finissant par un achèvement au passé composé, provenant de la seconde ("Je roulais dans la rue Pasteur quand une voiture surgit de ma droite"). (Ce groupe fournit en fait 45% des séquences verbales situées à cheval entre les deux parties). C'est la raison pour laquelle nous l'avons baptisé groupe des circonstances ou de la survenue d'un incident. Le récit est ici majoritairement en arrière-plan. L'acteur A ou un tiers se trouvent fortement représentés au détriment de l'acteur B. Il y a peu d'allusion aux causes de l'accident ou au choc, et peu de verbes en avant-plan. L'évocation de buts ou d'alternatives est absente de ce groupe (contrairement au précédent).

3) **Cluster (AA) des Actions menant à l'Accident**
Description. Le troisième groupe (AA) marque nettement le récit de l'accident proprement dit. Il est caractérisé par l'abondance des accomplissements, au détriment des états et des activités. Il est caractérisé par un mode perfectif, mais comprend beaucoup d'infinitif. Les prototypes détectés sont donnés Tableau 4.3. Ce type de construction se prête à des enchaînements, comme "j'ai voulu m'engager pour laisser", ou "n'ayant pas la possibilité de changer de voie et la route étant mouillée", qui permettent, par enchaînement, la sélection d'assez long segment de texte.

Interprétation. La plupart des séquences proviennent comme on l'a dit de la seconde partie du récit, et c'est pourquoi, ce groupe contenant aussi beaucoup d'accomplissements, nous l'avons nommé groupe des actions menant à l'accident.

Néanmoins, les participes présents et les infinitifs permettant, comme dans le groupe (C), l'expression de buts et de mondes possibles ("désirant me rendre à", "commençant à tourner"), certaines séquences peuvent parfois figurer dans la première partie du récit. On constate cependant une assez forte proportion d'acteurs A et B, et peu de tiers, mais également peu de relief - le récit n'étant ni spécialement en avant-plan, ni en arrière-plan. On trouve beaucoup de verbes participant à la chaîne causale de l'accident, mais relativement peu mentionnant le choc.

Synthèse	**verbe 1**	**verbe 2**
type 5	acc.(ou ach) INF	acc.(ou ach) INF (ou ppr)
type 6	ach. (ou acc.), PC	acc.(ou ach.) INF
type 7	états, PC	ach. INF

TAB. 4.3 – Synthèse des prototypes du cluster (AA)

4) **Cluster (CC) du Choc et des Commentaires**
Description. Les verbes d'achèvements (45%) figurent ici en plus grand nombre que partout ailleurs (32% en moyenne), au détriment des activités (seulement 6,5%), et des états (14,5% au lieu de 23,5%). Ceci atteste que ce groupe de transitions, comme le précédent, favorise globalement la description de l'accident. On observe ici une augmentation des infinitifs et des participes sur le premier verbe au détriment de l'imparfait et du présent, et une augmentation massive du passé composé sur le second au détriment de toutes les catégories - sauf le présent (8%, légèrement supérieur à la moyenne). Cette apparition du présent explique peut-être la forte proportion de commentaires (29%), plus importante qu'ailleurs (18% en moyenne). Le Tableau 4.4 donne deux des prototypes (peu nombreux dans ce groupe, plus difficile à caractériser). L'analyse en terme de point de vue montre que la séquence finit globalement par un point de vue perfectif.

Interprétation. Ce groupe est caractéristique de la description de l'accident proprement dit, et des commentaires sur ce dernier. La fin du récit est favorisée. La mention de but ou d'alternative y est moyenne. Par contre, c'est ici que l'avant-plan est le plus marqué. Il y a un nombre important d'acteur B (l'adversaire) et c'est ici que l'on trouve le plus de verbes relatifs aux causes de l'accident et au choc lui-même.

La table qui suit indique les principales caractéristiques sémantiques des groupes ob-

Synthèse	**verbe 1**	**verbe 2**
type 8	acc.(ou ach) INF	ach., PC
type 9	ach. (ou état), PC	ach.(ou acc.), PC

TAB. 4.4 – Synthèse des prototypes du cluster (CC)

tenus.

Cluster (CC) du choc et des commentaires	**Cluster (AA) des actions conduisant à l'accident**
causalité très forte, mise en avant, choc fréquent, buts et alternatives	causalité forte, relief neutre, quelques chocs, nombreux buts et alternatives
Cluster (CI) des circonstances ou de la survenue d'un incident	**Cluster (C) des circonstances**
peu de causes, arrière-plan, peu de choc, ni but ou alternative	pas de causalité, arrière-plan, pas de choc, nombreux buts et alternatives

TAB. 4.5 – Résumé des différenciations sémantiques

4.4 Conclusion

Dans ce chapitre nous avons décrit deux approches pour construire un modèle de mélange markovien topologique. Ils ont pour objectif la classification non-supervisée des données séquentielles. L'approche globale représente un modèle original qui pourrait être appliqué à des données plus complexes. Dans ce chapitre on a fourni le détail des formules mathématiques et nous avons développé les expressions permettant d'estimer les paramètres en utilisant l'algorithme EM avec l'algorithme Baum-Welch en prenant en compte d'une manière explicite la topologie du graphe associé au modèle. Des techniques de visualisation plus raffinés peuvent être développées en se basant sur cette approche pour illustrer le pouvoir du modèle global d'explorer les données non i.i.d.

Nous avons aussi présenté un cas particulier de l'approche globale que nous avons défini comme approche locale. Ce modèle est proche des modèles hybrides traditionnels, et il a permis d'allier les points forts des cartes topologiques probabilistes et les modèles des HMMs pour la classification non supervisée de données séquentielles. Chaque cluster est représenté par un sous graphe de la carte permettant une initialisation du modèle HMM. Cette structure est ensuite optimisée par apprentissage. L'optimisation des modèles HMM par apprentissage permet un élagage des états redondants de la carte topologique et par conséquent une adaptation de l'architecture aux données séquentielles.

Les apports essentiels des deux approches que nous avons proposé peuvent être résumés dans les points suivants :

- En effet, la première approche est un modèle unifiant les cartes topologiques et les chaînes de Markov. C'est un modèle unique où la topologie est utilisée d'une manière explicite dans le modèle HMM. La deuxième approche permet d'utiliser la capacité du codage topologique des données à travers la carte topologique, pour préparer les données d'apprentissage aux HMM.

- Elaboration d'une topologie des modèles HMM. Dans la première approche, on construit un HMM topologique sans aucun prétraitement des données. Par contre dans la deuxième approche la segmentation de l'espace des données par les cartes permet de définir une notion de voisinage entre les modèles HMM.

- Découverte par apprentissage de la structure des modèles HMM adaptée aux données structurées en séquence.

Conclusions et perspectives

Nous avons présenté dans ce mémoire le travail de recherche mené dans l'équipe A3 (Apprentissage Artificiel & Applications) du laboratoire LIPN (Laboratoire d'Informatique de Paris-Nord), UMR 7030 du CNRS. Les travaux présentés dans ce manuscrit nous ont permis d'analyser les propriétés mathématiques de l'algorithme des cartes auto-organisatrices dans un cadre probabiliste.

Ce type de modèle, introduit initialement par T. Kohonen dans le cas des données numériques, constitue toujours un thème actif de recherche dans le domaine de l'apprentissage non supervisé. Les cartes topologiques auto-organisatrices présentent un certain nombre d'intérêts. Elles réalisent une partition des données en groupes ou clusters "homogènes" organisés facilitant l'interprétation et la visualisation des données. L'organisation des données en une partition de plusieurs groupes “homogènes“ constitue une base à d'autres modèles de classification.

Tenir compte de la distribution des données implique l'intégration de plus d'informations dans le modèle des cartes topologiques selon le type de données et donc l'amélioration des performances. Nous avons proposé dans le cadre de ce travail de recherche trois nouveaux modèles d'apprentissage qui tiennent compte de la distribution locale des données. Les trois modèles s'inspirent des cartes topologiques probabilistes dédiées aux données continues.

Le premier modèle BeSOM (Bernoulli Self-Organizing Map) que nous avons proposé utilise un formalisme probabiliste dédié à la classification non supervisée des données binaires dans lequel les cellules de la carte sont représentées par une distribution de Bernoulli. Chaque cellule est caractérisée par un prototype avec le même codage binaire de l'espace d'entrée et par la probabilité d'être différent de ce prototype.

Le second modèle, PrMTM (Probabilistic Mixed Topological Map) dédié aux données mixtes (quantitatives et qualitatives), utilise le formalisme probabiliste des cartes

topologiques et associe simultanément à chaque cellule de la carte la distribution Gaussienne et la distribution de Bernoulli. Ce modèle est une extension du modèle BeSOM et PrSOM, et permet de respecter le codage mixte des données dans l'espace d'entrée.

Le troisième modèle est dédié aux données structurées en séquences. Deux approches sont présentées : la première est une reformulation des cartes topologiques avec des modèles de mélanges Markovien. Ce modèle permet de définir un nouvel algorithme d'apprentissage des chaines de Markov où la structure topologique du graphe associé est utilisée d'une manière implicite. La deuxième approche est un modèle hybride où les points forts des cartes topologiques probabilistes et les modèles Markoviens sont alliés pour la classification non supervisée de données séquentielles.

L'algorithme d'apprentissage qui est associé à tous les algorithmes proposés découle de l'application de l'algorithme EM sur une fonction de vraisemblance à maximiser. L'intérêt pratique de ces algorithmes réside dans le fait que l'extension des cartes topologiques auto-organisatrices aux données binaires, mixtes et séquentielles, permet d'étendre le champ d'application de celles-ci à des domaines très variés.

Plusieurs perspectives de développement peuvent être envisagées comme suite à ce travail :

- Dans le cadre de cette thèse nous avons traité les données binaires de type disjonctives, or il est possible d'étendre le modèle aux données qualitatives ordinales en utilisant la loi de Poisson.

- L'utilisation de la carte pour faire de la classification automatique nécessite de procéder à des regroupements des cellules obtenues. Ce regroupement peut se faire par une classification hiérarchique ascendante. Il serait intéressant de généraliser ce processus à différents types de données.

- Nous pouvons envisager éventuellement l'extension des modèles BeSOM et PrMTM à la classification croisée en utilisant le formalisme probabiliste des cartes topologiques. Nous pouvons envisager un critère entre groupes d'observations et groupes de variables qui découle de la fonction de vraisemblance.

- Dans le cadre du traitement de données structurées en séquences, l'assimilation des cartes topologiques probabilistes (BeSOM, PrMTM, PrSOM) aux chaînes de

Markov cachées, mérite d'être approfondie. L'intérêt de cette approche est qu'elle permet d'utiliser au mieux le formalisme probabiliste. Une extension vers un modèle "on-line" serait assez intéressante pour traiter de grandes masses de données séquentielles.

Annexes

Annexe A : Données binaires et qualitatives

Dans cette thèse on s'intéresse tout d'abord aux observations binaires c'est-à-dire aux observations $\mathbf{x} = (x^1, ..., x^n)$ dont les composants x^j sont des variables de $\beta = \{0, 1\}$, ainsi $\mathbf{x} \in \mathcal{A} \subset \beta^n$, [40, 41, 111]. Les variables booléenes codent les variables qualitativesà deux modalités. D'une manière générale, on s'intéressera aux observations $\mathbf{x}$ dont les composantes sont des variables qualitativesà plusieurs modalités. Souvent ce type de données est le résulat d'enquêtes ou de sondages, où les individus répondentà chaque question en choisissant une modalité et une seule parmi les modalités proposées pour cette question. Dans ce cas chaque réponse au questionnaire correspond à une observation qui peut être codée sous sa forme brute $\mathbf{x} = (x^1, ..., x^n)$ où $x^j \in A_j$ qui est l'ensemble fini formé par l'énumération des m_j modalités de la $j^{ième}$ question, dans ce cas $\mathbf{x} \in \mathcal{A} \subset A_1 \times A_2 \times .... \times A_n$.

Il existe en statistique deux manières de coder de telles observations sous la forme de vecteur binaire, [40] : Le codage disjonctif complet et le codage additif selon que la variable qualitative est ordinale ou disjonctive. On réécrit chaque observation (questionnaire donné) sous la forme d'un vecteur binaire (booléen) ; pour cela il faut coder chacune de ses composantes, qui est une variable qualitativeà plusieurs modalités en un vecteur binaire. Chaque variable qualitativeà m modalités est alors codée par un vecteur binaireà m composantes, le codage est différent suivant que la variable qualitative est ordinale ou disjonctive. Une variable **qualitative ordinale** est une variable dont ses m modalités sont régies par un ordre total implicite (exemple : petit, moyen, grand, très grand). Afin de conserver cette notion d'ordre, la variable peut être transformée en un vecteur binaire par **le codage binaire additif**. Ce qui permet essentiellement de rester cohérent avec la notion d'ordre entre les modalités. Formellement, si on veut coder une variable qualitativeà m modalités, la q^{ieme} modalité de cette variable va être

représentée par un vecteur de dimension m (nombre de modalités) dans lequel les q premières composantes sont égalesà 1 et les composantes restantes égalesà zéro.
Le codage disjonctif complet permet de coder en un vecteur binaire, une variable **qualitative disjonctive**, pour laquelle il n'existe aucune relation d'ordre entre ses m modalités (exemple : bleu, rouge, vert, blanc). Formellement, la q^{ieme} modalité de cette variable sera codée par un vecteur binaire de dimension m (nombre de modalités) dans lequel toutes les composantes sont nulles sauf la q^{ieme} composante qui est égaleà 1.

Ainsi, étant donné une observation $\mathbf{x}$ donnée à n composantes qualitatives (dimension des observations) et dont la j^{ieme} composante admet m_j modalités, chaque composante peut être codée, par le codage additif ou le codage disjonctif complet, en un vecteur binaire. Ainsi, cette observation sera codée globalement par un vecteur binaire de dimension $\sum_{j=1}^{n} m_j$, (le nombre total des modalités).
Dans l'espace des données binaires des mesures de similarités, appelées aussi distances binaires sont utilisées pour mesurer la ressemblance entre deux observations, $\mathbf{x}_1$ et $\mathbf{x}_2$ appartenantà l'espace β^n. Celles-ci sont calculéesà partir de la table de contingence des deux vecteurs binaires associés, il s'agit d'une matrice dont les éléments représentent le nombre d'occurences communes de 1 et de 0 et le nombre de désaccords d'individus ayant choisi les différents couples de modalités 1 et 0, table 6.

	1 $\mathbf{x}_1$ 0	
1	a	b
$\mathbf{x}_2$		
0	c	d

TAB. 6 – Table de contingence. a et d représentent respectivement le nombre de fois que les deux individus choisissent la même modalité "1" ou "0". b et c représentent le nombre de fois que le premier individu (le deuxième individu) choisit la modalité "1" et le deuxième individu (le premier individu) choisit la modalité "0"

On trouve dans la littérature plusieurs indices de similarités calculésà partir de la table de contingence, tel que l'indice Hamman, Jaccard, Kulezynski, Mountfird, Mzeley, Ochiai,Rogers, Russel, Simple matching coefficient, Yule, Hamming et Tanimoto, [41]. On représente queulques de ces indices dans le tableau 7. La distance de Hamming (18), représente le nombre de différences entre deux vecteurs binaires, elle sera notée par la suite par $\mathcal{H}$. Elle sera la distance utilisée pour tout nos modèles décrits dans la thèse.

Nom	Distance
Simple matching	$1 - \frac{a+d}{a+b+c+d}$
Jaccard	$\frac{b+c}{a+b+c}$
Russell et Rao	$1 - \frac{a}{a+b+c+d}$
Dice	$\frac{b+c}{2a+b+c}$
Rogers et Tanimoto	$\frac{2(b+c)}{a+2(b+c)+d}$
Pearson	$\frac{1}{2} \quad \frac{ad \quad bc}{2\sqrt{(a+b)(a+c)(d+b)(d+c)}}$
Yule	$\frac{bc}{ad+bc}$
Sokal Michener	$\frac{2b+2c}{a+2b+2c+d}$
Kulzinski	$\frac{2b+2c+d}{a+2b+2c+d}$
Hamming	$b+c$

TAB. 7 – Différents indices de similarité

Caractéristiques de la distance de Hamming

En statistique descriptive, il est possible de résumer un ensemble d'observations par des grandeurs caractéristiques. Dans le cas de la distance euclidienne, il est possible de résumer un ensemble réel par sa moyenne et son écart-type, ou si les observations sont en dimensions multiples par le centre de gravité et son inertie. Des caractéristiques équivalentes ont été définies pour la distance de Hamming, elles permettent de caractériser un ensemble de vecteurs binaires (en dimensions simples ou multiples) à l'aide d'une valeur centrale, elle-même binaire et d'un écart-type [111].

La distance de Hamming binaire entre $\mathbf{x}_1 = (x_1^1, ..., x_1^n)$ et $\mathbf{x}_2 = (x_2^1, ..., x_2^n)$ de β^n est égale au nombre de composantes différentes entre ces deux vecteurs. Elle est définie par la formule suivante :

$$\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2) = \sum_{j=1}^{n} |x_1^j - x_2^j| \tag{18}$$

Médiane Binaire

Soit x une variable binaire $x \in \{0, 1\}$ et A un ensemble de N observations, $A = \{x_i, i = 1..N\}$, pour laquelle chaque observation x_i est pondérée par γ_i, avec $\sum_i \gamma_i = 1$. Toute valeur w de $\{0, 1\}$ minimisant :

$$\mathcal{I}(w) = \sum_{i=1}^{N} \gamma_i |x_i - w| \tag{19}$$

est une caractéristique de la valeur centrale de la variable x, elle correspondà **la valeur médiane** définie par l'échantillon A. Etant donné que pour les données binaires on a $|x - w| = (1 - x)w + x(1 - w)$, l'expression (19) peut être écrite de la façon suivante :

$$\mathcal{I}(w) = \sum_{i=1}^{N} \gamma_i (1 - x_i) w + \sum_{i=1}^{N} \gamma_i x_i (1 - w)$$

$$\mathcal{I}(w) = w\Gamma_0 + (1 - w)\Gamma_1 \tag{20}$$

Formule pour laquelle $\Gamma_0 = \sum_{i=1}^{N} \gamma_i (1 - x_i)$, représente la somme des pondérations des observations de l'échantillon dont la valeur est égaleà 0, et $\Gamma_1 = \sum_{i=1}^{N} \gamma_i x_i$ la somme des pondérations des observations de l'échantillon dont la valeur est égaleà 1.

Trouver la valeur médiane w qui minimise $\mathcal{I}(w)$ revient doncà choisir w de façonà ce que $\mathcal{I}(w)$ ait comme valeur le minimum de Γ_0 et Γ_1. Ceci revient à prendre $w = 1$ si $\Gamma_1 > \Gamma_0$ et $w = 0$ si $\Gamma_1 < \Gamma_0$. Une ambiguïté se présente si $\Gamma_1 = \Gamma_0$. Lorsque les individus formant l'échantillon sont munis de la même pondération, la valeur médiane est tout simplement la valeur majoritaire de l'échantillon $\mathcal{A}$.

Remarque
Le calcul précédent montre bien que la valeur médiane d'un échantillon reste la même lorsqu'on multiplie toutes les pondérations par une même constante. Ainsi, la condition $\sum_{i=1}^{N} \gamma_i = 1$ n'est pas nécessaire pour le calcul de la médiane, par la suite et pour le calcul de la médiane, $\Gamma = \sum_{i=1}^{N} \gamma_i$ n'est pas nécessairement égaleà l'unité.

Centre Médian

Si l'on considère maintenant un ensemble de vecteurs binaires de dimension n, l'ensemble d'échantillons $\mathcal{A}$ est formé de vecteurs binaires, ($\mathcal{A} \subset \beta^n$). Comme précedemment, on suppose que chaque vecteur $\mathbf{x}_i$ de $\mathcal{A}$ est muni d'une pondération $\gamma_i \geq 0$. Etant donné un vecteur $\mathbf{w} = (w^1, w^2, ..., w^n) \in \beta^n$, on définit l'**inertie** des vecteurs de $\mathcal{A}$ par rapportà $\mathbf{w}$ par l'expression suivante :

$$\mathcal{I}(\mathbf{w}) = \sum_{i=1}^{N} \gamma_i \mathcal{H}(\mathbf{x}_i, \mathbf{w}) = \sum_{i=1}^{N} \gamma_i \sum_{j=1}^{n} |x_i^j - w^j| \tag{21}$$

Cette notion d'inertie permet de définir la valeur centrale d'un ensemble d'observations $\mathcal{A}$, appellé **centre médian**. Le centre médian est défini comme le vecteur binaire $\mathbf{w}$ qui minimise l'inertie $\mathcal{I}(\mathbf{w})$, (21). Cette inertie se décompose comme suit :

$$\mathcal{I}(\mathbf{w}) = \sum_{j=1}^{n} \mathcal{I}(w^j)$$

avec

$$\mathcal{I}(w^j) = \sum_{i=1}^{N} \gamma_i |x_i^j - w^j|$$

Minimiser $\mathcal{I}(\mathbf{w})$ par rapport au vecteur binaire $\mathbf{w} = (w^1, ..., w^j, ..., w^n)$ revientà minimiser $\mathcal{I}(w^j)$ sur chacune de ses composantes. j ($1 \leq j \leq n$). D'après ce qui précède, la valeur w^j recherchée est la valeur médiane de l'échantillon $\{x_i^j, i = 1..N\}$ pondérée par les pondérations $\{\gamma_i, i = 1..N\}$. Ainsi pour calculer la j^{ieme} composante w^j du centre médian on doit calculer $\Gamma_0^j = \sum_{x_i^j=0} \gamma_i$ et $\Gamma_1^j = \sum_{x_i^j=1} \gamma_i$

$$w^j = \begin{cases} 0 \text{ si } \Gamma_0^j \geq \Gamma_1^j \\ 1 \text{ si } \Gamma_1^j > \Gamma_0^j \end{cases},$$

Remarque
Lorsque toutes les pondérations γ_i sont égalesà 1, la composante w^j est alors la valeur 0 ou 1 la plus souvent choisie (majoritaire) par les individus sur la variable j.

Si l'on considère l'exemple donné dans les tables 8 et 9 dans lesquelles on a codé trois ensembles $\mathcal{A}_1, \mathcal{A}_2$ et $\mathcal{A}_3$ de 6 observations représentant une variable qualitativeà 3, 5, et 4 modalités, on observe que dans le cas du codage additif, la valeur médiane représente toujours une des modalités de la variable qualitative considérée. Dans le cas du codage disjonctif complet, la médiane peut ne pas être interprétable. Le risque dans ce cas est de trouver des centres médians qui ne représentent aucune modalité. En aucun cas, ils ne représentent des modalités contradictoires.

individus/Modalités	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
$\mathbf{x}_1$	110	11000	1000
$\mathbf{x}_2$	100	10000	1111
$\mathbf{x}_3$	111	11111	1100
$\mathbf{x}_4$	111	11000	1110
$\mathbf{x}_5$	100	11000	1111
$\mathbf{x}_6$	111	11100	1110
médiane **w**	**111**	**11000**	**1110**

TAB. 8 – Tableau des variables catégorielles codées en additif ; on remarque que dans les trois cas la médiane représente l'une des modalités de la variable

individus/Modalités	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
$\mathbf{x}_1$	010	01000	1000
$\mathbf{x}_2$	100	10000	0001
$\mathbf{x}_3$	001	00001	0100
$\mathbf{x}_4$	001	01000	0010
$\mathbf{x}_5$	100	01000	0001
$\mathbf{x}_6$	001	00100	0010
médiane $\mathbf{w}$	001	01000	**0000**

TAB. 9 – Tableau des variables catégorielles codées en disjonctif complet ; la médiane de l'ensemble $\mathcal{A}_3$ est vide et ne représente aucune modalité.

Liste des publications

1) (soumis 2009) ROGOVSCHI N., LEBBAH M., and BENNANI Y. Learning Self-Organizing Mixture Markov Models. Journal of Nonlinear Systems and Applications.
2) ROGOVSCHI N., LEBBAH M., BENNANI Y. (2009), "Un algorithme pour la classification topographique simultanée de données qualitatives et quantitatives", CAp 09 : Conférence francophone sur l'apprentissage automatique. p209-224. Plate-forme AFIA, 25-29 Mai, Hammamat-Tunisie.
3) LEBBAH M., BENNANI Y., ROGOVSCHI N. (2009), «Learning Self-Organizing Maps as a Mixture Markov Models», Proc. of the ICCSA'09, p54-59, The 3rd International Conference on Complex Systems and Applications, Le Havre, Normandy, France, June 29 - July 02, 2009.
4) GROZAVU N., and ROGOVSCHI N. (2009) "Mining visual data", ICMCS, Moldova, octobre 2009.
5) ROGOVSCHI N., LEBBAH M., and BENNANI Y. (2008). Probabilistic Mixed Topological Map for Categorical and Continuous Data. Machine Learning and Applications. ICMLA 2008 : Seventh International Conference. pp 224-231. San Diego, California, December 11-13, 2008.
6) LEBBAH M., BENNANI Y., ROGOVSCHI N. (2008). A Probabilistic Self-Organizing Map for Binary Data Topographic Clustering. International Journal of Computational Intelligence and Applications 7(4) : 363-383 (2008).
7) LEBBAH M., ROGOVSCHI N., and BENNANI Y. (2007). BeSOM. : Bernoulli on Self Organizing Map. International Joint Conferences on Neural Networks". p631-636 IJCNN 2007-August 12-17, 2007, Orlando, Florida.
8) LEBBAH M., ROGOVSCHI N., BENNANI Y. (2007), "BeSOM : Bernoulli on Self-Organizing Map", Cap 2007 : conférence francophone sur l'apprentissage automatique, Plate-forme AFIA, 2-6 juillet, Grenoble.
9) ROGOVSCHI N., VIENNET E., (2007). "Finding community structure in social networks using statistical approaches", ICMCS, Moldova, septembre 2007.

10) ROGOVSCHI N., VIENNET E. (2007). “Finding community structure in large social networks. Poster at NATO Advanced Study Institute on Mining Massive Data Sets for Security”, September 2007.

11) RECANATI C., ROGOVSCHI N., BENNANI Y., « Hybrid Unsupervised Learning to Uncover Discourse Structure», Responding to Information Society Challenges : New Advances in Human Language Technologies, Zygmunt Vetulani et Hans Uszkoreit (eds), LNAI vol 5603, 2009, pp. 258-269.

12) RECANATI C. et ROGOVSCHI N., “Enchaînements verbaux "étude sur le temps et l'aspect utilisant des techniques d'apprentissage non supervisé ”. Actes de la 14ème conférence sur le Traitement Automatique des Langues Naturelles, juin 2007, IRIT Press, Toulouse, pp. 379-388.

13) ROGOVSCHI N., BENNANI Y., RECANATI C.(2007). “Apprentissage neuromarkovien pour la classification non supervisée de données structurées en séquences”. Actes des 7èmes journées francophones Extraction et Gestion des Connaissances, Atelier Fouille de données et algorithmes biomimétiques, jan 2007, Namur, Belgique.

14) RECANATI C., ROGOVSCHI N., BENNANI Y. (2007), "Sequencing of verbs" a study on tense and aspect using methods of unsupervised learning, Proceedings of RANLP'07, International Conference on Recent Advances in NLP, Sept. 27-29, 2007, Borovets, Bulgaria.

15) RECANATI C., ROGOVSCHI N., BENNANI Y. (2007). The structure of verbal sequences analyzed with unsupervised learning techniques”, Proceedings of The 3rd Language and Technology Conference : Human Language Technologies as a Challenge for Computer Science and Linguistics, Oct. 5-7, 2007, Poznan, Pologne. (Selected among the best papers of LTC'07)

Bibliographie

[1] Amir Ahmad and Lipika Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data Knowl. Eng.*, 63(2) :503–527, 2007.

[2] Edoardo M Airoldi. Getting started in probabilistic graphical models. *PLoS Comput Biol*, 3(12) :e252, 12 2007.

[3] Bill Andreopoulos, Aijun An, and Xiaogang Wang. Bi-level clustering of mixed categorical and numerical biomedical data. *International Journal of Data Mining and Bioinformatics*, 1(1) :19 – 56, 2006.

[4] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics : ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2) :49–60, 1999.

[5] F. Anouar. Modélisation probabiliste des auto-organisées : Application en classification et en régression. thèse de doctorat soutenue au conservatoires national des arts et métiers. 1996.

[6] Fatiha Anouar, Fouad Badran, and Sylvie Thiria. Self-organizing map, a probabilistic approach. In *Proceedings of WSOM'97-Workshop on Self-Organizing Maps, Espoo, Finland June 4-6*, pages 339–344, 1997.

[7] P. ARABIE and L.J. HUBERT. An overview of combinatorial data analysis. *in : Arabie, P., Hubert, L.J., and Soete, G.D. (Eds.) Clustering and Classification*, pages 5–63, 1996.

[8] Khalid Benabdeslem Arnaud Zeboulon, Younès Bennani. Hybrid connectionist approach for knowledge discovery from web navigation patterns. In *ACS/IEEE International Conference on Computer Systems and Applications*, pages 118–122, 2003.

[9] Michaël Aupetit. Learning topology with the generative gaussian graph and the em algorithm. In *NIPS*, pages 592–598, 2005.

[10] G. BALL and D. HALL. Isodata, a novel method of data analysis and classification. In *Technical Report AD-699616, SRI, Stanford, CA.*, 1965.

[11] J. BANFIELD and A. RAFTERY. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49 :803–821, 1993.

[12] Daniel Barbará and Ping Chen. Using the fractal dimension to cluster datasets. In *KDD '00 : Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 260–264, New York, NY, USA, 2000. ACM.

[13] O. Barndorff-Nielsen. Information and exponential families in statistical theory. *Wiley, Chichester*, 1978.

[14] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3 :1–8, 1972.

[15] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree : an efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2) :322–331, 1990.

[16] Amir Ben-Dor and Zohar Yakhini. Clustering gene expression patterns. In *RECOMB '99 : Proceedings of the third annual international conference on Computational molecular biology*, pages 33–42, New York, NY, USA, 1999. ACM.

[17] Samy Bengio and Yoshua Bengio. An EM algorithm for asynchronous input/output hidden Markov models. In L. Xu, editor, *International Conference On Neural Information Processing*, pages 328–334, Hong-Kong, 1996.

[18] Yoshua Bengio and Paolo Frasconi. An input output hmm architecture. In *NIPS*, pages 427–434, 1994.

[19] P. BERKHIN and J. BECHER. Learning simple relations : Theory and applications. In *In Proceedings of the 2nd SIAM ICDM.*

[20] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[21] C. M. Bishop. Neural networks for pattern recognition. *Clarrendon Press, Oxford*, 1995.

[22] C. M. Bishop, M. Svensén, and C. K. I.Williams. Gtm : The generative topographic mapping. *Neural Comput*, 10(1) :215–234, 1998.

[23] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), 2006.

[24] C.L. Blake and C.L Merz. Uci repository of machine learning databases. technical report. 1998.

[25] H.H. BOCK. Probability models in partitional cluster analysis. In *In Ferligoj, A. and Kramberger, A. (Eds.) Developments in Data Analysis*, pages 3–25, Slovenia, 1996.

[26] L. BOTTOU and Y. BENGIO. Convergence properties of the k-means algorithms. In *In Tesauro, G. and Touretzky, D. (Eds.) Advances in Neural Information Processing Systems 7*, pages 585–592, The MIT Press, Cambridge, MA., 1995.

[27] D. Bouchaffra and J. Tan. Structural hidden markov models : An application to handwritten numeral recognition. *Intell. Data Anal.*, 10(1) :67–79, 2006.

[28] Djamel Bouchaffra. Embedding hmm's-based models in a euclidean space : The topological hidden markov models. In *ICPR08*, pages 1–4, 2008.

[29] Djamel Bouchaffra and Jun Tan. Structural hidden markov model and its application in automotive industry. In *ICEIS (2)*, pages 155–164, 2003.

[30] Herve A. Bourlard and Nelson Morgan. *Connectionist Speech Recognition : A Hybrid Approach*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[31] H. BOZDOGAN. Mixture-model cluster analysis using model selection criteria and a new information measure of complexity.

[32] H. BOZDOGAN. Determining the number of component clusters in the standard multivariate normal mixture model using model-selection criteria. 49, 1983.

[33] P.S. Bradley, Usama Fayyad, and Cory Reina. Scaling clustering algorithms to large databases. pages 9–15. AAAI Press, 1998.

[34] Igor V. Cadez, Padhraic Smyth, and Heikki Mannila. Probabilistic modeling of transaction data with applications to profiling, visualization, and prediction. In *KDD '01 : Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 37–46, New York, NY, USA, 2001. ACM.

[35] Gilles Celeux and Gérard Govaert. Clustering criteria for discrete data and latent class models. *Journal of classification*, (8) :157–176, 1991.

[36] P. Cheeseman and J. Stutz. Bayesian classification (AUTOCLASS) : Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press/MIT Press, 1996.

[37] Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman. Autoclass : A bayesian classification system. In *Fifth International Workshop on Machine learning*, pages 54–64, 1988.

[38] Tom Chiu, DongPing Fang, John Chen, Yao Wang, and Christopher Jeris. A robust and scalable clustering algorithm for mixed type attributes in large database

environment. In *KDD '01 : Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–268, New York, NY, USA, 2001. ACM.

[39] Thomas M. Cover and Joy A. Thomas. *Elements of information theory.* Wiley-Interscience, New York, NY, USA, 1991.

[40] D R. Cox. The analysis of binary data. Chapman and Hall, 1970.

[41] M. A. Cox and T. F. Cox. Mutidimensional scaling. Chapman and Hall, 1994.

[42] Douglas R. Cutting, Jan O. Pedersen, David R. Karger, and John W. Tukey. Scatter/gather : A cluster-based approach to browsing large document collections. In Nicholas J. Belkin, Peter Ingwersen, and Annelise Mark Pejtersen, editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992*, pages 318–329. ACM, 1992.

[43] William H. Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. volume 1, pages 7–24, December 1984.

[44] D. Defays. An efficient algorithm for a complete link method. *Comput. J.*, 20(4) :364–366, 1977.

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

[46] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Roy. Statist. Soc*, 39(1) :1–38, 1977.

[47] FAN J. DHILLON, I. and Y. GUAN. Efficient clustering of very large document collections. *In Grossman, R.L., Kamath, C., Kegelmeyer, P., Kumar, V., and Namburu, R.R. (Eds.)*, 2001.

[48] Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar. Enhanced word clustering for hierarchical text classification. In *KDD '02 : Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 191–200, New York, NY, USA, 2002. ACM.

[49] Inderjit S. Dhillon and Dharmendra S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Revised Papers from Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD*, pages 245–260, London, UK, 2000. Springer-Verlag.

[50] Davies D.L. and Bouldin D.W. A cluster separation measure. In *IEEE, Transactions on Pattern Analysis and Machine Learning*, volume 1, 1979.

[51] R. DUDA and P. HART. Pattern classification and scene analysis. 1973.

[52] L. ENGLEMAN and J. HARTIGAN. Percentage points of a test for clusters. *Journal of the American Statistical Association*, (64) :1647–1648, 1969.

[53] KRIEGEL H-P. ESTER, M. and X. XU. A database interface for clustering in large spatial databases. In *In Proceedings of the 1st ACM SIGKDD*, pages 94–99. Montreal, Canada., 1995.

[54] KRIEGEL H-P. SANDER J. ESTER, M. and X. XU. A density-based algorithm for discovering clusters in large spatial databases with noise. In *In Proceedings of the 2nd ACM SIGKDD*, pages 226–231, Portland, Oregon, 1996. IEEE Computer Society.

[55] Martin Ester, Alexander Frommelt, Hans-Peter Kriegel, and Jöorg Sander. Spatial data mining : Database primitives, algorithms and efficient dbms support. *Data Min. Knowl. Discov.*, 4(2-3) :193–216, 2000.

[56] Recanati C. et Recanati F. La classification de vendler revue et corigée. In *Cahiers Chronos 4, La modalité sous tous ses aspects*, 1999.

[57] B. EVERITT. *Cluster Analysis (3rd ed.). Edward Arnold, London, UK.* Edward Arnold, London, UK., 1993.

[58] Christos Ferles and Andreas Stafylopatis. A hybrid self-organizing model for sequence analysis. In *ICTAI '08 : Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence*, pages 105–112, Washington, DC, USA, 2008. IEEE Computer Society.

[59] Christos Ferles and Andreas Stafylopatis. Sequence clustering with the self-organizing hidden markov model map. In *BIBE*, pages 1–7. IEEE, 2008.

[60] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2) :139–172, 1987.

[61] Douglas H. Fisher. Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research. (JAIR)*, 4 :147–178, 1996.

[62] E. Forgy. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*, 21 :768–780, 1965.

[63] C. FRALEY and A. RAFTERY. Mclust : Software for model-based cluster and discriminant analysis. In *Tech Report 342*. Dept. Statistics, Univ. of Washington., 1999.

[64] Chris Fraley, Adrian, and E. Raftery. How many clusters ? which clustering method ? answers via model-based cluster analysis. *The Computer Journal*, 41 :578–588, 1998.

[65] B. Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA, 1995.

[66] B. Fritzke. Self-organizing network that can follow non-stationary distributions. *Proceedings of ICANN'97, " International Conference on Artificial Neural Networks, Springer*, 1997.

[67] Bernd Fritzke. Kohonen feature maps and growing cell structures - a performance comparison. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 123–130, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[68] K. FUKUNAGA. Introduction to statistical pattern recognition. In *Academic Press, San Diego, CA*, 1990.

[69] E.H. Han G. Karypis and V. Kumar. Multilevel refinement for hierarchical clustering. In *Technical Report TR-99-020*, Department of Computer Science, University of Minnesota, Minneapolis, 1999.

[70] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artif. Intell.*, 40(1-3) :11–61, 1989.

[71] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden markov models. *Mach. Learn.*, 29(2-3) :245–273, 1997.

[72] M. Girolami. The topographic organisation and visualisation of binary data using mutivariate-bernoulli latent variable models. *I.E.E.E Transactions on Neural Networks*, 12(6) :1367–1374, 2001.

[73] T.F. GONZALEZ. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(3) :293–306, 1985.

[74] T. Graepel, M. Burger, and K. Obermayer. Self-organizing maps : generalizations and new optimization techniques. *Neurocomputing*, 21 :173–190, 1998.

[75] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure : An efficient clustering algorithm for large databases. In *SIGMOD Conference*, pages 73–84, 1998.

[76] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock : A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*, pages 512–521. IEEE Computer Society, 1999.

[77] Spath H. Clustering algorithms. *John Wiley and Sons, New York, NY*, 1975.

[78] Spath H. Cluster analysis algorithms. *Ellis Horwood, Chichester, England*, 1980.

[79] J. HAN and M. KAMBER. Data mining. In *Morgan Kaufmann Publishers*, 2001.

[80] KAMBER M. HAN, J. and A. K. H. TUNG. Spatial clustering methods in data mining : A survey. In *In Miller, H. and Han, J. (Eds.) Geographic Data Mining and Knowledge Discovery, Taylor and Francis*, 2001.

[81] J. A. Hartigan and M. A. Wong. Algorithm AS 136 : A k-means clustering algorithm. *Applied Statistics*, 28(1) :100–108, 1979.

[82] J. HEER and E. CHI. Identification of web user traffic composition using multi-modal clustering and information scent. *1-st SIAM ICDM, Workshop on Web Mining*, pages 51–58, 2001.

[83] T. Heskes. Self-organizing maps, vector quantization, and mixture modeling. *IEEE Trans. Neural Networks*, 12 :1299–1305, 2001.

[84] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 58–65, 1998.

[85] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Min. Knowl. Discov.*, 2(3) :283–304, 1998.

[86] Lynette Hunt and Murray Jorgensen. Mixture model clustering using the multimix program. *Austral. & New Zealand J Statistics*, 41(2) :153–171, 1999.

[87] RISSANEN J. Stochastic complexity in statistical inquiry. In *World Scientific Publishing Co.*, pages 3–25, Singapore, 1989.

[88] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering : a review. *ACM Computing Surveys*, 31(3) :264–323, 1999.

[89] A.K. Jain and P.J. Flynn. "image segmentation using clustering,". In *Advances in Image Understanding : A Festschrift for Azriel Rosenfeld*, pages 65–83, 1996.

[90] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[91] A. Kaban and M. Girolami. A combined latent class and trait model for the analysis and visualization of discrete data. *IEEE Trans. Pattern Anal. Mach. Intell*, 23 :859–872, 2001.

[92] Eser Kandogan. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In *KDD '01 : Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 107–116, New York, NY, USA, 2001. ACM.

[93] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon : Hierarchical clustering using dynamic modeling. *Computer*, 32(8) :68–75, 1999.

[94] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1) :359–392, 1998.

[95] R. KASS and A. RAFTERY. Bayes factors. In *Journal of Amer. Statistical Association*, volume 90, 1995.

[96] L. Kaufman and P. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley-Interscience, New York, 1990.

[97] Shehroz S. Khan and Shri Kant. Computation of initial modes for k-modes clustering algorithm using evidence accumulation. In *IJCAI*, pages 2784–2789, 2007.

[98] T. Kohonen. *Self-organizing Maps*. Springer Berlin, 2001.

[99] T. Kohonen. *Self-organizing Maps*. Springer Berlin, Vol. 30, Springer, Berlin, Heidelberg, New York, 1995, 1997, 2001. Third Extended Edition, 501 pages, 2001.

[100] E. KOLATCH. Clustering algorithms for spatial databases : A survey. 2001.

[101] G. LANCE and WILLIAMS W. A general theory of classification sorting strategies. *Computer Journal*, 9 :373–386, 1967.

[102] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *KDD '99 : Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, New York, NY, USA, 1999. ACM.

[103] M. lebbah. *Carte topologique pour données qualitatives : application à la reconnaissance automatique de la densité du trafic routier*. Thése de doctorat en Informatique, Université de Versailles Saint-Quentin en Yvelines, 2003.

[104] M. Lebbah, S. Thiria, and F. Badran. Topological map for binary data. In *Proceedings European Symposium on Artificial Neural Networks-ESANN 2000, Bruges, April 26-27-28*, pages 267–272, 2000.

[105] Mustapha Lebbah, Aymeric Chazottes, Fouad Badran, and Sylvie Thiria. Mixed topological map. In *ESANN*, pages 357–362, 2005.

[106] Mustapha Lebbah, Nicoleta Rogovschi, and Younés Bennani. Besom : Bernoulli on self organizing map. In *International Joint Conferences on Neural Networks. IJCNN 2007, Orlando, Florida, August 12-17*, 2007.

[107] C. Lin and M. Chen. On the optimal clustering of sequential data. pages 141–157, 2002.

[108] S. P Luttrel. A bayesian ananlysis of self-organizing maps. *Neural Computing*, 6 :767 – 794, 1994.

[109] Jorgensen M.A. and Hunt L.A. Mixture model clustering of data sets with categorical and continuous variables. In Eds. D. L. Dowe K. B. Korb J. J. Oliver World Scientific, editor, *The conference, Information, Statistics and Induction in Science*, pages 375 – 384. MIT Press, 1996.

[110] J. MAO and A.K. JAIN. A self-organizing network for hyperellipsoidal clustering (hec). *IEEE Transactions on Neural Networks*, 7(1) :16–29, 1996.

[111] F. Marchetti. Contribution à la classification de données binaires et qualitatives. In *thèse de l'université de Metz*, 1989.

[112] J.L. MARROQUIN and F. GIROSI. Some extensions of the k-means algorithm for image segmentation and pattern classification. In *KDD '00 : Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, MIT, Cambridge, MA., 1993. A.I. Memo 1390.

[113] K J. Martinetz, T. Schulten. "a neural gas", network learns topologies in artificial neural network. *T.Kohonen, K. Makisara, O. Simula et J.Kangas, eds*, pages 397–402, 1991.

[114] D. MASSART and L. KAUFMAN. The interpretation of analytical chemical data by the use of cluster analysis. 1983.

[115] G. McLachlan and T. Krishman. *The EM algorithm and Extensions*. Wiley, New York, 1997.

[116] G. J. McLachlan and K. E. Basford. *Mixture Models, Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

[117] Marina Meila and Michael I. Jordan. Learning fine motion by markov mixtures of experts. Technical report, Cambridge, MA, USA, 1995.

[118] G. MILLIGAN and M. COOPER. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, (50) :159–179, 1985.

[119] B. MIRKIN. *Mathematic Classification and Clustering*. Kluwer Academic Publishers, 1996.

[120] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1997.

[121] Andrew Moore. Very fast em-based mixture model clustering using multiresolution kd-trees. In *In Advances in Neural Information Processing Systems 11*, pages 543–549. MIT Press, 1998.

[122] Andrew W. Moore. Very fast em-based mixture model clustering using multiresolution kd-trees. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 543–549, Cambridge, MA, USA, 1999. MIT Press.

[123] F. Murtagh. Multidimensional clustering algorithms. In *COMPSTAT Lectures 4*, Wuerzburg, 1985. Physica-Verlag.

[124] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *Comput. J.*, 26(4) :354–359, 1983.

[125] M. Nadif and G. Govaert. Clustering for binary data and mixture models : Choice of the model. *Applied Stochastic Models and Data Analysis*, 13 :269–278, 1998.

[126] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 144–155. Morgan Kaufmann, 1994.

[127] S. Oja, E. Kaski. *Kohonen maps*. Elsevier, 1999.

[128] Jonathan J. Oliver, Rohan A. Baxter, and Chris S. Wallace. Unsupervised learning using mml. In *ICML*, pages 364–372, 1996.

[129] Clark F. Olson. Parallel algorithms for hierarchical clustering. volume 21, pages 1313–1325, 1995.

[130] Dan Pelleg and Andrew Moore. Accelerating exact k-means algorithms with geometric reasoning. In *KDD '99 : Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 277–281, New York, NY, USA, 1999. ACM.

[131] Dan Pelleg and Andrew Moore. X-means : Extending k-means with efficient estimation of the number of clusters. In *Proc. ICML*, pages 727–734, San Francisco, 2000. Morgan Kaufmann.

[132] Rodolphe Priam and Mohamed Nadif. Carte auto-organisatrice probabiliste sur données binaires. In *EGC*, pages 445–456, 2006.

[133] Rodolphe Priam, Mohamed Nadif, and Gérard Govaert. Binary block gtm : Carte auto-organisatrice probabiliste pour les grands tableaux binaires. In *Extraction et gestion des connaissances (EGC'2008), Actes des 8èmes journées Extraction et Gestion des Connaissances*, Revue des Nouvelles Technologies de l'Information, pages 265–272, Sophia-Antipolis, France,, 29 janvier au 1er février 2008. Cépaduès-Éditions.

[134] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2) :257–286, 1989.

[135] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336) :846–850, 1971.

[136] J. RISSANEN. Modeling by shortest data description. In *Automatica*, volume 14.

[137] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases : The algorithm gdbscan and its applications. *Data Min. Knowl. Discov.*, 2(2) :169–194, 1998.

[138] E. Schikuta. Grid-clustering : An efficient hierarchical clustering method for very large data sets. In *ICPR '96 : Proceedings of the 13th International Conference on Pattern Recognition*, page 101, Washington, DC, USA, 1996. IEEE Computer Society.

[139] Erich Schikuta and Martin Erhart. The bang-clustering system : Grid-based data analysis. In *IDA*, pages 513–524, 1997.

[140] G. SCHWARZ. Estimating the dimension of a model. 6 :461–464, 1978.

[141] D.W. SCOTT. Multivariate density estimation. In *Wiley, New York, NY*, 1992.

[142] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. Wavecluster : a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal*, 8(3-4) :289–304, 2000.

[143] R. Sibson. Slink : An optimally efficient algorithm for the single-link cluster method. *Comput. J.*, 16(1) :30–34, 1973.

[144] Padhraic Smyth. Model selection for probalistic clustering using cross-validation likelihood. Technical Report ICS-TR-98-09, 1998.

[145] Panu Somervuo. Competing hidden markov models on the self-organizing map. *Neural Networks, IEEE - INNS - ENNS International Joint Conference on*, 3 :3169, 2000.

[146] Panu Somervuo and Teuvo Kohonen. Self-organizing maps and learning vector quantization forfeature sequences. *Neural Process. Lett.*, 10(2) :151–159, 1999.

[147] Myra Spiliopoulou and Carsten Pohle. Data mining for measuring and improving the success of web sites. *Data Min. Knowl. Discov.*, 5(1-2) :85–114, 2001.

[148] KARYPIS G. STEINBACH, M. and V. KUMAR. A comparison of document clustering techniques. In *6-th ACM SIGKDD, World Text Mining Conference,Boston, MA*, 2000.

[149] Gascuel O. Canu S. Thiria S., Lechevallier Y. Statistique et méthodes neuronales. 1997.

[150] J.J. Verbeek, N. Vlassis, and B.J.A. Kröse. Self-organizing mixture models. *Neurocomputing*, 63 :99–123, 2005.

[151] Ellen M Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Inf. Process. Manage.*, 22(6) :465–476, 1986.

[152] C. WALLACE and D. DOWE. Intrinsic classification by mml - the snob program. *In the Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*, pages 37–44, 1994.

[153] C. WALLACE and P. FREEMAN. Estimation and inference by compact coding. In *Journal of the Royal Statistical Society, Series B*, volume 49, pages 240–265, 1987.

[154] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting : A statistical information grid approach to spatial data mining. In *VLDB*, pages 186–195, 1997.

[155] J.H. WARD. Hierarchical grouping to optimize an objective function. *Journal Amer. Stat, Assoc.*, 58(301) :235–244, 1963.

[156] Xiaowei Xu, Martin Ester, Hans-Peter Kriegel, and Jörg Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the Fourteenth International Conference on Data Engineering, February 23-27, 1998, Orlando, Florida, USA*, pages 324–331. IEEE Computer Society, 1998.

[157] Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4) :721–736, 1982.

[158] Vendler Z. Verbes and times. In *Linguistics in Philosophy*, 1967.

[159] B. ZHANG. Generalized k-harmonic means - dynamic weighting of data in unsupervised learning. In Proceedings of the 1st SIAM ICDM, Chicago, IL, 2001.

Résumé

Le travail de recherche exposé dans cette thèse concerne le développement d'approches à base de cartes auto-organisatrices dans un formalisme de modèles de mélanges pour le traitement de données qualitatives, mixtes et séquentielles. Pour chaque type de données, un modèle d'apprentissage non supervisé adapté est proposé. Le premier modèle, décrit dans cette étude, est un nouvel algorithme d'apprentissage des cartes topologiques BeSOM (Bernoulli Self-Organizing Map) dédié aux données binaires. Chaque cellule de la carte est associée à une distribution de Bernoulli. L'apprentissage dans ce modèle a pour objectif d'estimer la fonction densité sous forme d'un mélange de densités élémentaires. Chaque densité élémentaire est-elle aussi un mélange de lois Bernoulli définies sur un voisinage. Le second modèle aborde le problème des approches probabilistes pour le partitionnement des données mixtes (quantitatives et qualitatives). Le modèle s'inspire de travaux antérieurs qui modélisent une distribution par un mélange de lois de Bernoulli et de lois Gaussiennnes. Cette approche donne une autre dimension aux cartes topologiques : elle permet une interprétation probabiliste des cartes et offre la possibilité de tirer profit de la distribution locale associée aux variables continues et catégorielles. En ce qui concerne le troisième modèle présenté dans cette thèse, il décrit un nouveau formalisme de mélanges Markovien dédiés au traitement de données structurées en séquences. L'approche que nous proposons est une généralisation des chaines de Markov traditionnelles. Deux variantes sont développées : une approche globale où la topologie est utilisée d'une manière implicite et une approche locale où la topologie est utilisée d'une manière explicite. Les résultats obtenus sur la validation des approches traités dans cette étude sont encourageants et prometteurs à la fois pour la classification et pour la modélisation.

Mots clés : apprentissage non-supérvisé, cartes auto-organisatrices, modèles de mélanges, chaînes de Markov, données catégorielles, données mixtes, données séquentielles.

Abstact

The research presented in this thesis concerns the development of self-organising map approaches based on mixture models which deal with different kinds of data : qualitative, mixed and sequential. For each type of data we propose an adapted unsupervised learning model. The first model, described in this work, is a new learning algorithm of topological map BeSOM (Bernoulli Self-Organizing Map) dedicated to binary data. Each map cell is associated with a Bernoulli distribution. In this model, the learning has the objective to estimate the density function presented as a mixture of densities. Each density is as well a mixture of Bernoulli distribution defined on a neighbourhood. The second model touches upon the problem of probability approaches for the mixed data clustering (quantitative and qualitative). The model is inspired by previous works which define a distribution by a mixture of Bernoulli and Gaussian distributions. This approach gives a different dimension to topological map : it allows probability map interpretation and offers the possibility to take advantage of local distribution associated with continuous and categorical variables. As for the third model presented in this thesis, it is a new Markov mixture model applied to treatment of the data structured in sequences. The approach that we propose is a generalisation of traditional Markov chains. There are two versions : the global approach, where topology is used implicitly, and the local approach where topology is used explicitly. The results obtained upon the validation of all the methods are encouraging and promising, both for classification and modelling.

Key words : unsupervised learning, self-organised-maps, mixture models, hidden Markov models, mixed data, sequential data, categorical data.