Estimation robuste des modèles de mélange sur des données distribuées

Ali El Attar

UNIVERSITÉ DE NANTES

UFR DES SCIENCES ET TECHNIQUES

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE MATHÉMATIQUES

Année 2012

N° attribué par la bibliothèque

Estimation robuste des modèles de mélange sur des données distribuées

---

THÈSE DE DOCTORAT
Discipline : INFORMATIQUE
Spécialité : INFORMATIQUE

Présentée
et soutenue publiquement par

Ali EL ATTAR

le 17 juillet 2012 au LINA, devant le jury ci-dessous

Président	:	_____	_____
Rapporteurs	:	Franck MORVAN, Professeur	Université Paul Sabatier
		Mustapha LEBBAH, Maître de conférences-HDR	Université Paris XIII
Examinateurs	:	Pierre-François MARTEAU, Professeur	Université de Bretagne Sud
		Colin DE LA HIGUERA, Professeur	Université de Nantes

Directeur de thèse : Marc GELGON
Co-encadrant de thèse : Antoine PIGEAU

ED : .....
(Uniquement pour STIM et SPIGA)

ESTIMATION ROBUSTE DES MODÈLES DE MÉLANGE SUR DES DONNÉES DISTRIBUÉES

Robust estimation of mixture model on distributed data

Ali EL ATTAR

favet neptunus eunti

Université de Nantes

Ali EL ATTAR

Estimation robuste des modèles de mélange sur des données distribuées

x+139 p.

Ce document a été préparé avec LaTeX2e et la classe `these-LINA` version v. 1.30 de l'association de jeunes chercheurs en informatique LoGiN, Université de Nantes. La classe `these-LINA` est disponible à l'adresse :

`http://login.lina.sciences.univ-nantes.fr/`

Impression : these.tex – 2/10/2012 – 0:55

Révision pour la classe : these-LINA.cls,v 1.30 2005/08/16 17:01:41 mancheron Exp

Résumé

Cette thèse propose une contribution en matière d'analyse de données, dans la perspective de systèmes informatiques distribués non-centralisés, pour le partage de données numériques. De tels systèmes se développent en particulier sur internet, possiblement à large échelle, mais aussi, par exemple, par des réseaux de capteurs. Notre objectif général est d'estimer la distribution de probabilité d'un jeu de données distribuées, à partir d'estimations locales de cette distribution, calculées sur des sous-jeux de données locaux. En d'autres termes, il s'est agi de proposer une technique pour agréger des estimés locaux pour en faire un estimé global. Notre proposition s'appuie sur la forme particulière que doivent prendre toutes les distributions de probabilité manipulées : elles doivent se formuler comme un mélange de lois gaussiennes multivariées. Notre contribution est une solution à la fois décentralisée et statistiquement robuste aux modèles locaux aberrants, pour mener à bien l'agrégation globale, à partir d'agrégations locales de mélanges de lois gaussiennes. Ces agrégations locales ne requièrent un accès qu'aux seuls paramètres des modèles de mélanges, et non aux données originales.

Mots-clés : clustering ; modèle de mélange, agrégation des modèles des mélanges ; estimation robuste ; détection de données atypiques ; données distribuées.

Abstract

This work proposes a contribution aiming at probabilistic model estimation, in the setting of distributed, decentralized, data-sharing computer systems. Such systems are developing over the internet, and also exist as sensor networks, for instance. Our general goal consists in estimating a probability distribution over a data set which is distributed into subsets located on the nodes of a distributed system. More precisely, we are at estimating the global distribution by aggregating local distributions, estimated on these local subsets. Our proposal exploits the following assumption: all distributions are modelled as a Gaussian mixture. Our contribution is a solution that is both decentralized and statistically robust to outlier local Gaussian mixture models. The proposed process only requires mixture parameters, rather than original data.

Keywords: clustering; mixture of models; aggregation of mixture models; robust estimation; outlier detection; distributed data.

Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse, Monsieur Marc GELGON, Professeur de l'Université de Nantes qui m'a dirigé durant ce travail. Je lui suis reconnaissant pour ses encouragements, son enthousiasme et sa confiance.

Je voudrais également remercier Monsieur Antoine PIGEAU, Maître de conférences de l'Université de Nantes pour m'avoir encadré, conseillé ainsi que pour ses corrections de ce mémoire.

Je remercie cordialement M. Franck MORVAN, Professeur de l'Université Paul Sabatier et M. Mustapha LEBBAH, Maître de conférences-HDR de l'Université Paris XIII d'avoir accepté d'être rapporteurs de cette thèse. Je souhaite adresser également mes remerciements à M. Pierre-François MARTEAU, Professeur de l'Université de Bretagne Sud et M. Colin DE LA HIGUERA, Professeur de l'Université de Nantes d'avoir fait l'honneur d'examiner mon travail.

J'adresse mes remerciements à tous les membres de l'équipe ATLAS-GRIM : les enseignants - chercheurs et mes collègues doctorants pour tous les échanges techniques, scientifiques et pour leur sympathie, leur accueil chaleureux pendant ces trois ans de thèse.

Pour le meilleur, c'est surtout une profonde pensée pour mon papa décédé, et qui aurait été fier de moi. Merci également à ma maman, ma soeur et mes frères pour leur amour, leur soutien sans faille et à tout ce qu'ils ont pu m'apporter pour franchir les obstacles les plus difficiles.

Sommaire

CHAPITRE 1

Introduction générale

Cette thèse aborde le problème de l'estimation statistique d'une distribution de probabilité dans $\mathbb{R}^n$, à partir de données réparties [46, 42, 114, 102]. L'hypothèse faite dans ce travail, qui oriente fortement les solutions proposées, est que les distributions de probabilité considérées, prennent la forme de modèles de mélanges gaussiens [98, 45, 20]. Nous avons cherché à définir un procédé réalisant une estimation statistiquement robuste de cette distribution de probabilité, notamment dans le contexte où l'estimation est conduite de manière décentralisée.

1.1 Contexte : sources de données multiples, modèles probabilistes de mélanges

Ce travail s'inscrit dans l'axe plus général des recherches visant, à exploiter les **masses de données mises à disposition sur des dépôts en ligne**, reliées par des réseaux à plus ou moins large échelle, au moyen de techniques d'analyse de données[4, 53, 2]. Plusieurs contextes motivent cette orientation générale, principalement les réseaux de capteurs et les données sur la toile. Discutons plus en détail le second cas. D'un réseau de documents textuels puis incluant des éléments audiovisuels, la toile a évolué vers un "web de données" plus complexe [30, 55]. En particulier, aux documents s'ajoutent actuellement de manière fortement croissante des jeux de données accessibles directement, ou via des API (croissance forte des "données ouvertes" prises dans une acception large, réseaux et média sociaux, jeux de données scientifiques, culturels ou de consommation,...). Sans que cela en soit un facteur exclusif, ce développement a bénéficié de divers mécanismes incitatifs de crowdsourcing (ou, plus largement, d'interactions entre producteurs, données/documents et consommateurs, interactions constituant elles-mêmes des données). Le caractère "participatif" (du point de vue utilisateur) du mécanisme alimentant certains jeux de données sur la toile, ne reflète pas nécessairement le degré de décentralisation de l'architecture du système informatique et du contrôle sous-jacent. Nombre de systèmes, toutefois, visant principalement le partage de données, mais aussi certains réseaux sociaux, privilégient une **architecture décentralisée** (avec des variantes dans le degré de décentralisation, par exemple selon les réseaux pair-à-pair). De bonnes propriétés en termes de passage à l'échelle, de tolérance aux fautes, d'usage de la bande passante et de maîtrise des données à caractère personnel sont souvent invoquées pour motiver cette orientation.

Ainsi, on observe une tendance à étendre la toile d'un réseau de documents vers une immense base de données globale. Si la résolution du problème de l'articulation entre les jeux de données composant cette dernière perspective reste généralement assez préliminaire, il nous semble qu'il s'agit d'une évolution franche et centrale, que faciliteront par exemple les progrès opérationnels des technologies de "Linked-Data". C'est une occasion de reconsidérer certains problèmes de fouille de données ou d'en identifier de

nouveaux, à la lumière des particularités de ce contexte. Cette voie bénéficie actuellement d'une forte dynamique de recherche, à la croisée de plusieurs communautés (fouille de données et apprentissage automatique, bases de données, systèmes répartis....). Un objectif intéressant consiste en la découverte d'une structure cachée (latente) présente dans un jeu de données global, composé d'une multitude de jeux de données locaux. Notre travail se place dans cette orientation, avec toutefois des hypothèses particulières et des objectifs bien plus restreints, que nous précisons ci-après.

Ce travail de thèse se place dans la perspective de multiples sources de données, dont on souhaite construire une description statistique globale [119, 13, 77, 35]. Pour simplifier, nous n'avons considéré que le cas où les données sont de même nature (des vecteurs de $\mathbb{R}^n$), correspondant au même espace d'attributs. En particulier, l'estimation de la distribution de probabilités des données est un objectif important, qui sera le nôtre, car riche de débouchés possibles. De façon générale, une des manières de mener à bien cette tâche d'estimation de la distribution de probabilité de données est de **modéliser cette distribution comme un mélange de lois de probabilités élémentaires** [88, 49, 124]. C'est l'approche retenue dans cette thèse. Les mélanges de lois constituent une démarche classique et puissante, permettant de modéliser des distributions de probabilité d'une manière à la fois flexible (une grande diversité de distributions peuvent être modélisées sous cette forme) et parcimonieuse (la distribution ainsi modélisée est décrite par un nombre réduit de paramètres) [19, 18, 86]. Par ailleurs, la composition de la distribution comme un mélange peut souvent être exploitée pour identifier une structure cachée dans le jeu de données, c.a.d. de déterminer des sous-populations intéressantes du point de vue de l'interprétation du jeu de données - c'est la vision "clustering" des mélanges de lois. Il existe une multitude de modèles de mélanges, caractérisés par des lois élémentaires (Gaussiennes, distributions de Student, Von Mises, multinomiales,...) adaptées à la nature des données, au phénomène à modéliser ou à des propriétés que l'on souhaite donner à l'estimation. Dans cette thèse, on considèrera le cas des mélanges gaussiens et des mélanges de lois de Student [28, 93]. Enfin, des variantes structurelles dans le modèle probabiliste de mélange existent, qui permettent par exemple de traiter conjointement le problème de clustering et de réduction de dimension de l'espace d'attributs (co-clustering).

1.2 Objectif, contributions et plan du document

Au croisement des sources de données distribuées et des modèles de mélanges probabilistes, ce travail de thèse a cherché à proposer une manière d'estimer un modèle global à partir de données distribuées [9, 10]. Plus précisément, on suppose un ensemble de sources de données, chacune permettant de réaliser localement l'estimation d'un modèle de mélange, sur les données locales. Cette hypothèse de situation étant donnée, **notre travail a consisté à faire "collaborer" cet ensemble de modèles probabilistes locaux**. Notre proposition a cherché à vérifier les propriétés suivantes :

- elle doit être décentralisée et asynchrone, fonctionnant par un ensemble d'interactions locales entre sources de données, n'ayant recours qu'à des informations locales ;

- elle doit être statistiquement robuste, à la fois à des données aberrantes localement et à des modèles locaux qui seraient globalement aberrants ;

– elle doit n'avoir recours, autant que possible, qu'aux paramètres des modèles probabilistes locaux estimés, plutôt qu'aux données locales. S'en tenir aux paramètres des modèles doit permettre une économie d'informations à manipuler (calculer, transmettre d'un site à l'autre) et ouvre peut-être des perspectives en terme de préservation de confidentialité des données individuelles [80]. Ce dernier point n'a toutefois été l'objet d'aucun examen précis au cours de cette thèse.

La contribution de cette thèse se compose de deux parties principales :

– **Estimation et agrégation locale robuste.**
Dans un premier temps, nous avons souhaité estimer des modèles de mélange de type gaussien, par un procédé rendant l'estimation peu sensible à la présence, dans le jeu de données, d'une minorité de données aberrantes [8]. Dans cet objectif, nous avons eu recours à des **mélanges de lois de Student**. En effet, substituer une distribution de Student à une distribution gaussienne permet de conférer à l'estimation de cette dernière, une certaine **robustesse statistique**. Cette propriété est visible quand on formule la distribution de Student comme un **mélange infini** de lois gaussiennes. Dans un second temps, nous traitons la question de l'agrégation de deux modèles de mélange de Student, estimés sur deux jeux de données supposés engendrés par le même phénomène générateur de données, un seul mélange, ne décrivant l'union des deux jeux de données pré-cités qu'avec le nombre nécessaire de composantes. La solution que nous proposons consiste en un schéma itératif s'appuyant sur la décomposition infinie évoquée des lois de Student. **Dans le chapitre 2**, nous dressons un panorama des techniques de clustering de données et d'estimation de modèles de mélange. **Dans le chapitre 3**, nous présentons la manière de rendre robuste ces tâches et proposons une technique permettant l'agrégation de modèles de mélange de type "Student".

– **Estimation globale robuste de mélange à partir de mélanges locaux.**
Dans cette seconde partie de notre travail, nous avons proposé une technique permettant d'estimer un modèle de mélange global, par succession d'agrégation de modèles locaux [9, 10]. Elle s'inspire de travaux développés dans le cadre des systèmes de propagation par rumeur. Le cas prototypique simple consiste à calculer la moyenne d'un ensemble de scalaires, de façon décentralisée. Notre tâche étend ce but en deux sens : (1) les éléments manipulés ne sont pas des scalaires, mais des distributions de probabilités prenant la forme de mélanges de lois (2) l'estimation doit être robuste à des modèles locaux aberrants (car estimés à partir de données aberrantes au vu du jeu de données global, ou estimés d'une manière non fiable. **Le chapitre 4** expose des éléments d'état de l'art concernant la classifiation non supervisée de données distribuées. **Le chapitre 5** présente ensuite notre proposition pour la tâche que nous venons d'énoncer. Son volet expérimental forme **le chapitre 6**.

Le chapitre 7 récapitule les points principaux du travail et en évoque quelques perspectives.

CHAPITRE 2

Un état de l'art sur les méthodes de clustering

2.1 Introduction

Dans ce chapitre, nous réalisons un état de l'art sur l'apprentissage non supervisé, en particulier sur la tâche de clustering [48, 1, 36, 12, 121, 96, 64]. Nous nous focalisons sur la technique de clustering probabiliste : les données traitées sont supposées générées à partir d'une distribution probabiliste. Nous allons introduire pour cela les modèles de mélange, ainsi que leur utilisation pour réaliser le clustering de données.

Rappelons que la motivation derrière l'utilisation de modèles de mélange (chapitre 1), est exprimée par le fait que la technique de clustering utilisant ce genre de modèles présente des intérêts pertinents, lorsqu'il est transposé dans des architectures distribuées, comme réseau Pair-à-Pair [114, 46, 42]. Cela peut être soit pour des raisons de respect d'anonymat [80], ou pour réduire la quantité des informations transmises dans le réseau (seuls les paramètres de modèles sont communiqués).

Tout d'abord, nous allons définir et présenter les concepts de la tâche de clustering. Puis nous allons montrer les principaux types de méthodes de clustering étudiées dans la littérature, ainsi que les problèmes et les limites associés. Nous nous concentrons sur les méthodes probabilistes. Pour cela, nous introduisons les modèles de mélange, et les principes de leur estimation. Nous détaillons aussi l'algorithme le plus connu dans ce domaine pour estimer les modèles de mélange : l'algorithme *EM*. Enfin, nous allons étudier le problème des données atypiques. Nous citons pour cette raison plusieurs techniques, étudiées pour répondre à ce genre de problème.

2.2 Tâche de clustering

Dans cette section nous nous intéressons à l'analyse de données utilisant les modèles prédictifs : étant donné un ensemble de données observées, le but est de prédire le comportement des données non observées. Cette tâche est appelée aussi apprentissage. Il faut bien distinguer l'apprentissage supervisé (comme la classification) de l'apprentissage non supervisé (comme le clustering).

La classification implique que les données soient étiquetées, en supposant a priori que les groupes existent (classes). Le problème peut être posé de la manière suivante : considérons l'existence d'un ensemble d'individus X et un autre ensemble Y regroupant les étiquettes associées à chaque individu. Il

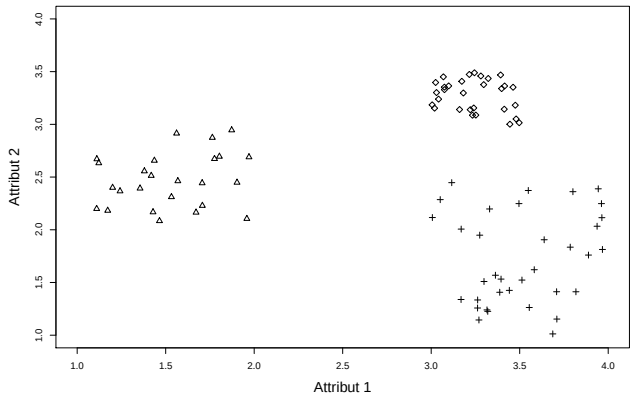


Figure 2.1 – Exemple d'un jeu de données décrites par deux attributs et contenant trois clusters identifiables visuellement.

consiste à apprendre une fonction (modèle) f entre X et Y, telle que $Y = f(X)$.

La procédure de classification supervisée est composée donc de deux étapes : en première étape, un modèle est construit à partir d'un ensemble d'exemples étiquetés par leur classe (phase d'apprentissage). Dans la deuxième étape, les étiquettes des nouveaux objets peuvent être prédites à partir de modèle préalablement appris.

Le clustering implique seulement les données non étiquetées sans aucune information a priori. Il sert à regrouper les objets ayant les mêmes propriétés. Chaque groupe (nommé cluster) doit vérifier deux propriétés principales : une similarité élevée entre ses objets, et une similarité faible avec les autres clusters. Un exemple d'un jeu de données décrites par deux attributs est illustré dans la figure (2.1). Il est clair visuellement que ces données forment 3 classes identifiables.

Il existe aussi l'apprentissage semi-supervisé qui utilise un ensemble de données étiquetées et non étiquetées. Il se situe entre l'apprentissage supervisé (toutes les données sont étiquetées) et l'apprentissage non-supervisé (données non-étiquetées). Il consiste à construire un classifieur en se basant sur les données étiquetées et à exploiter les données sans étiquettes pour améliorer les performances.

Après avoir présenté d'une manière très générale les techniques d'apprentissage (supervisée, non supervisée, semi-supervisée), nous nous concentrons par la suite sur la tâche de clustering. Nous allons montrer d'abord quelles sont les différentes propriétés à examiner sur les algorithmes de clustering.

2.3 Critère de choix d'un algorithme de clustering

Dans le domaine de clustering, il existe très peu de connaissances a priori sur la structure interne de données. Il est donc difficile, même pour un expert, de faire un choix parmi les nombreuses méthodes

disponibles. Plusieurs études ont été réalisées afin d'identifier le meilleur algorithme qui peut satisfaire en même temps, l'ensemble des axiomes de bases comme : type de données, formes de clusters, données atypiques [1]. Toutes ces études sont arrivées à la même conclusion : aucun algorithme de clustering n'est meilleur que les autres. Dans la suite, nous allons décrire un ensemble de propriétés liées aux algorithmes de clustering :

- manipulation des données d'entrée (numériques, catégoriques ou mixte) ;
- fonctionnement avec une grande masse de données (passage à l'échelle) ;
- découvrement de clusters avec de formes arbitraires. Celle-ci forme la propriété la plus importante pour choisir une des méthodes de clustering, mais il n'est pas facile de trouver un algorithme qui peut découvrir toutes les formes, en particulier si les données sont catégoriques ;
- traitement des données de grandes dimensions ;
- gestion du problème de présence de données atypiques dans le jeu de données en études ;
- utilisation d'un nombre limité de paramètres définis par les utilisateurs (c'est-à-dire les paramètres en entrée). Ces paramètres peuvent influencer directement les résultats de clustering comme par exemple le nombre de clusters, l'initialisation de clusters, la définition des seuils ;
- réalisation de clustering *dur* ou *flou*. Les méthodes de clustering *dur* consistent à attacher chaque élément à un seul cluster. Les méthodes hiérarchiques correspondent à ce type de clustering [50, 72]. En revanche, les méthodes de clustering *flou* permettent au même objet, d'être dans plusieurs clusters avec des degrés d'appartenance différents. Le degré d'appartenance d'un objet varie entre 0 et 1. Pour un objet et un cluster donné , plus le degré est proche de 1, plus la tendance d'être liée à ce cluster est forte. Ce type d'algorithmes est très utile dans les cas où les clusters sont chevauchés. Les méthodes probabilistes sont des exemples de méthodes de clustering *flou* [28]. Il est toujours possible d'obtenir un clustering *dur* à partir de clustering *flou*, en associant chaque objet, au cluster dont le degré d'appartenance est maximal. Un exemple d'application de clustering *dur* et *flou* est illustré dans la figure (2.2).

Dans cette section, nous avons présenté les propriétés à vérifier pour choisir un algorithme de clustering. Nous présentons ensuite une vue globale sur les différents types de méthodes de clustering étudiées dans la littérature. Nous détaillons en particulier les méthodes probabilistes, qui sont utilisées comme des méthodes de base pour réaliser le clustering dans notre thèse.

2.4 Les différentes techniques de clustering

Il n'est pas facile de présenter un état de l'art exhaustif de toutes les méthodes existantes. Cependant, nous décrivons dans cette section les principaux types d'approches de clustering. Elles sont regroupées selon des caractéristiques communes. Nous pouvons distinguer : les méthodes hiérarchiques, les méthodes basées sur la densité de points, les méthodes par partitionnement. Notre objectif est de mettre en

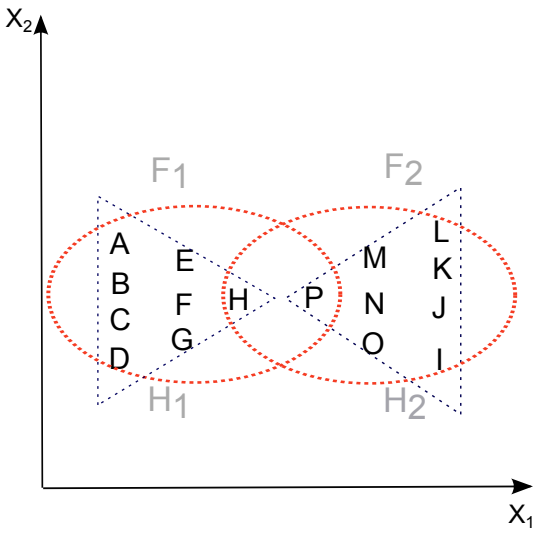


Figure 2.2 – Cette figure montre la différence entre le clustering *dur* et *flou*. Selon la technique de clustering *dur*, l'ensemble des données présentées dans la figure, est classé en deux clusters (triangles en bleus) $H_1 = \{$A,B,C,D,E,F,G,H$\}$, et $H_2 = \{$I,J,K,L,M,N,O,P$\}$. Chaque objet appartient à un seul cluster. En revanche, un algorithme de clustering *flou* peut produire les deux clusters $F_1 = \{$A,B,C,D,E,F,G,H,P$\}$, et $F_2 = \{$H,I,J,K,L,M,N,O,P$\}$, illustrés par les ellipses rouges. Dans le clustering *flou*, chaque objet a un degré d'appartenance à chaque cluster. Par exemple, H peut avoir le degré 0.6 dans F_1, et 0.4 dans F_2. Le clustering *dur* peut être réalisé toujours à partir des résultats de clustering *flou*. Dans notre exemple, H sera associé au cluster F_1 en *dur* (selon le degré d'appartenance le plus élevé).

valeur les différentes propriétés de chaque approche et de pouvoir ainsi motiver notre choix. Nous présentons rapidement les différentes techniques. Mais nous détaillons en particulier les modèles probabilistes (un type de méthode par partitionnement). La motivation derrière l'utilisation des modèles probabilistes (ou modèles de mélange) a été expliquée dans le chapitre (1). Nous commençons d'abord par présenter d'une manière très générale notre technique de réduction des modèles de mélange.

2.4.1 Vue générale de notre technique de clustering

Le but de notre approche est de réduire un modèle de mélange en un autre modèle avec un nombre plus petit de composantes. Les résultats des processus de réduction est donc un modèle de mélange (ou modèle de clusters) en regroupant les composantes similaires dans le modèle initial. Nous supposons d'abord que ce modèle est estimé à partir d'un ensemble de données en appliquant un algorithme d'estimation probabiliste (comme *EM* [28]). Ensuite l'algorithme de réduction est effectué en itérant entre ces deux étapes :

– calculer la distance entre les composantes de modèle initial et réduit afin de trouver les plus proches entre elles ;

– mettre à jour les paramètres du modèle réduit selon les résultats obtenus dans l'étape précédente ;

Notre approche est réalisée donc d'une façon itérative. Elle est appliquée à des composantes de modèles plutôt que des données. Dans la suite nous présentons les différentes techniques de clustering en mettant en valeur leurs propriétés.

2.4.2 Clustering hiérarchique

Les méthodes hiérarchiques [127, 50, 72, 51], consistent à construire une hiérarchie de clusters, c'est-à-dire un arbre binaire de clusters, nommé dendrogramme (figure 2.3). Ce dendrogramme illustre comment les clusters sont liés entre eux. Il permet de représenter les changements dans les clusters par plusieurs niveaux de granularité. Un noeud i à un niveau donné contient ses clusters enfants (fusionnement de ces deux clusters pour former le cluster de i ou division de cluster i en deux clusters enfants) et constitue une partition des objets de ses clusters parents. Les méthodes hiérarchiques sont classées en deux types d'approches, les approches ascendantes et les approches descendantes.

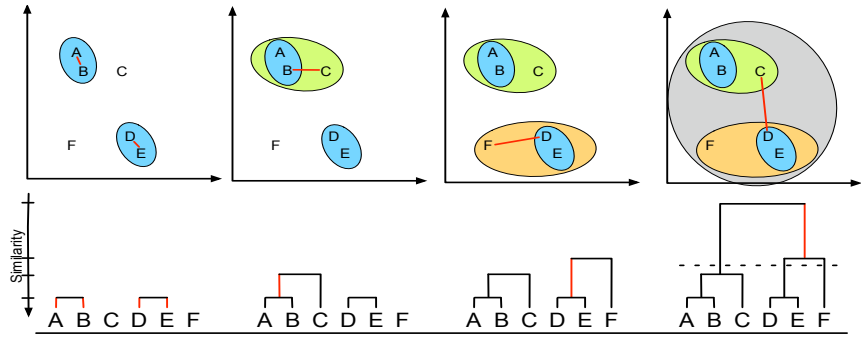


Figure 2.3 – Cette figure présente un dendrogramme d'une méthode hiérarchique ascendante. Dans chaque niveau hiérarchique, les clusters les plus proches sont fusionnés.

Les approches hiérarchiques ascendantes démarrent par le bas de la hiérarchie et se terminent par un seul cluster contenant tous les objets d'un jeu de données. Elles supposent au départ que chaque objet constitue un cluster. Ensuite, dans chaque étape, les deux clusters les plus proches entre eux sont fusionnés en un seul cluster. Cette opération continue jusqu'à avoir un seul cluster à la fin.

Les méthodes hiérarchiques descendantes suivent la stratégie inverse. Elles commencent par un seul cluster qui contient tous les objets d'un jeu de données (haut de la hiérachie), et se divisent successivement en clusters avec un nombre plus petit de données. Cette opération est répétée jusqu'à ce que chaque objet se trouve dans un seul cluster.

La fusion ou la division des clusters dans les méthodes hiérarchiques dépend directement du critère de (dis)similarité utilisé entre les classes. Pour choisir les clusters à fusionner ou à diviser, une matrice de distance des clusters deux à deux est utilisée. Il existe plusieurs stratégies pour calculer la similarité entre les clusters, dont les plus connues sont : *lien simple* (paire d'objets les plus proches), *lien complet* (paire des plus éloignés) et *lien moyen* (distance moyenne entre toutes les paires d'objets). Un exemple

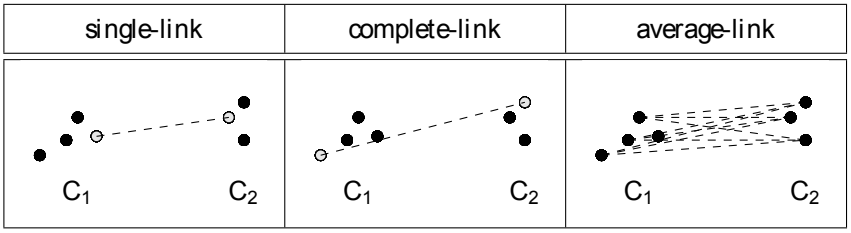


Figure 2.4 – Cette figure montre 3 stratégies différentes pour calculer la distance entre 2 clusters dans les méthodes hiérarchiques. La technique *single-link* définit la distance entre deux clusters comme la distance minimale des paires d'objets des deux groupes. Au contraire, la stratégie *complete-link* consiste à calculer la distance la plus grande entre des paires d'objets. Enfin, la méthode *average-link* cherche à calculer la moyenne des distances entre toutes les paires d'objets de deux clusters.

1 **Initialisation :**
2 - chaque élément du jeu de données est placé dans son propre cluster ;
3 - calculer la matrice de similarité M entre chaque couple de clusters (ici entre 2 éléments), en utilisant une de stratégies existantes pour calculer la similarité entre les clusters ;
4 **répéter**
5 - sélectionner à partir de M les deux clusters les plus proches C_I et C_J ;
6 - fusionner C_I et C_J pour obtenir un cluster C_G plus général comportant les éléments de deux clusters ;
7 - mettre à jour la matrice M en calculant la similarité entre C_G et les clusters existants ;
8 **jusqu'à** *la fusion de deux derniers clusters ou un critère d'arrêt est satisfait*;

Algorithme 1: Algorithme générale de méthodes hiérarchiques ascendantes

de ces trois stratégies est illustré dans la figure (2.4). L'algorithme (1) représente les étapes principales des méthodes de clustering hiérarchiques ascendantes.

Dans les méthodes hiérarchiques, il est possible de définir un critère d'arrêt de l'algorithme au delà duquel on ne fusionne (ou divise) plus les clusters. Ce critère se base sur la similarité entre les clusters. Donc les clustering hiérarchiques n'imposent pas de préciser a priori le nombre de clusters. Il est possible d'effectuer une coupe dans le dendrogramme à un niveau donné pour obtenir le nombre de clusters souhaité. Pour cette raison un certain nombre d'heuristiques sont proposés dans la littérature pour obtenir le meilleur partitionnement de données [64].

Les méthodes hiérarchiques possèdent plusieurs avantages, mais aussi des limites. Dans la suite, nous décrivons ici, quelques principaux avantages et inconvénients :

– elles fournissent des partitions à différents niveaux de granularité. Un critère d'arrêt peut être utilisé pour arrêter l'algorithme. Cependant, il est parfois difficile de préciser ces critères, et par conséquent d'avoir la meilleure partitionnement des données ;

– elles utilisent des mesures de similarité appliquées généralement sur des données numériques. Elles limitent donc la découverte de clusters aux formes convexes [94] ;

– elles ont une complexité de calcul élevée : si n est le nombre de données, alors elle est de l'ordre $O(n^2)$ pour le lien simple et $O(n^2log(n))$ pour le lien complet et moyen. Elles sont aussi besoin d'un espace important en mémoire, surtout si le nombre de données est important [94] ;

– elles réalisent un clustering *dur* en imposant à toutes les données d'appartenir à un des clusters (même les données atypiques). Elles sont donc sensibles aux données atypiques ;

– il est impossible de faire des améliorations des clusters (arbres de noeuds) une fois construits. Ceci est due à l'affectation statique des données (c'est-à-dire une fois qu'une donnée est affectée à un cluster, elle reste dans le même cluster tout au long de la construction de la hiérarchie).

Plusieurs limites sont donc liées aux méthodes de clustering hiérarchiques comme : données atypiques, formes arbitraires de clusters, complexité de calcul élevée. Plusieurs techniques ont été proposées pour répondre à ce genre de problèmes et pour améliorer la qualité de leurs résultats. Par exemple, les algorithmes *BIRCH* et *CURE* [127, 50] ont étudié le problème de données atypiques (données qui n'appartiennent à aucun cluster, ou qui si elles appartiennent à un cluster, contiennent un nombre très petit de données), et également le problème du clustering d'un nombre très large de données. Les travaux de *BIRCH* et *CHAMELEON* [50, 72] ont montré de plus, des techniques pour identifier les différentes formes, densités, et tailles de clusters. La méthode de *CHAMELEON* [72] est la plus évoluée parmi ces méthodes. Cependant, ses résultats dépendent directement du choix des paramètres définis par l'utilisateur.

Les méthodes hiérarchiques peuvent fournir plusieurs partitionnement de données (représentation sur plusieurs niveaux de granularité). Elles peuvent être considérées comme des approches génériques dans le domaine du clustering. En d'autres termes, elles peuvent être appliquées sur n'importe quel jeu de données en définissant la mesure de similarité entre les objets. Nous pouvons également réaliser un clustering hiérarchique dans notre cas, en regroupant chaque fois les deux modèles les plus proches entre eux. Cependant nous cherchons à améliorer d'une manière itérative notre clustering plutôt que de construire une hiérarchie de clustering.

2.4.3 Clustering basé sur la densité des points

Dans ce type d'algorithmes, les clusters sont identifiés en fonction de la densité de points dans les régions. Les régions de densités de points élevés représentent les clusters normaux, tandis que les régions de densités de points faibles se réfèrent aux clusters de données atypiques ou isolées. Un cluster est représenté par un ensemble de données connectées entre elles selon une certaine densité. La figure (2.5) montre un exemple de clustering basé sur la densité de points.

Plusieurs techniques basées sur la densité des points, ont été proposées dans la littérature [60, 6, 122, 105], dont la plus connue est *DBSCAN*, étudiée dans [32]. L'idée principale de ces techniques consiste à calculer le voisinage de chaque objet, c'est-à-dire les objets situés à une certaine distance de celui-ci. Si un objet possède un nombre important de voisinage, alors il est considéré comme appartenant à une

région dense, sinon, il est déclaré comme une donnée atypique. La majorité de ces méthodes nécessitent la déclaration de deux paramètres principaux : le rayon de voisinage autour des points, et le nombre minimal de points qui doit être trouvé dans ce voisinage.

Notons que les approches par grille peuvent être incluses dans les méthodes basées sur la densité [3]. Elles consistent à décomposer l'espace de données en un ensemble de cellules. Les cellules contenant un certain nombre de points et dépassant un seuil donné, sont considérées comme denses. Les cellules denses les plus proches, sont regroupées entre elles pour former des clusters.

Les algorithmes de clustering basés sur les densités sont capables d'identifier les différentes formes de clusters. En outre, ils fournissent des protections naturelles contre les données atypiques. Ils sont généralement appliqués pour classer les données numériques de faibles dimensions, connues sous le nom de données spatiales. Cependant, ils sont très sensibles aux paramètres définis par l'utilisateur (rayon de voisinage).

Ce type d'approches est difficile à appliquer dans notre contexte, notamment, la notion de connectivité entre les modèles puisqu'elle dépend de paramètres (nombre de voisins et rayon de voisinage) qui doivent être fixés manuellement. Par exemple, il n'est pas facile de trouver une valeur pertinente pour initialiser un rayon de voisinage d'un ensemble des modèles de mélange. Ainsi les approches basées sur la densité nécessitent une vue globale de données (modèles dans notre cas). Cette contrainte rend leurs applications dans les environnements distribués, où les modèles de données sont réparties sur plusieurs sites, plus complexes.

2.4.4 Clustering par partitionnement

Les méthodes par partitionnement consistent à trouver le clustering (meilleure partition) de données d'une façon itérative en minimisant un critère donné (comme l'erreur quadratique) [63, 113, 104, 62, 11, 84]. Les méthodes par partitionnement itèrent entre deux phases : une phase d'affectation de données aux clusters et une phase d'estimation de paramètres. Elles peuvent être divisées en méthodes probabilistes (les données sont supposées générées à partir d'une distribution probabiliste), et non probabilistes (comme *K-means*, *K-medoids*).

La complexité des modèles dans les méthodes par partitionnement, dépend de l'approche utilisée (probabiliste ou non probabiliste). Dans les méthodes probabilistes, une composante (cluster) d'un modèle de mélange gaussien par exemple, est définie par son centre, sa variance et une proportion dans le mélange. Dans *K-means*, chaque composante est représentée par la moyenne de ses données. La méthode *K-medoids* utilise la donnée la plus représentative pour définir la composante. Les paramètres de modèle sont ensuite déterminés en optimisant une fonction d'erreur (erreur quadratique comme dans *K-means* [63] ou fonction de vraisemblance comme dans l'algorithme probabiliste *EM* [28]).

Une solution naïve consiste à trouver la meilleure partition parmi toutes les combinaisons possibles de partitions. Cependant, cette solution peut avoir une complexité de calcul élevée. Il est donc impossible de l'appliquer en pratique, surtout si le nombre d'objets est important. Par exemple, il y a $2 * 10^{14}$ façons de partitionner 20 objets entre 3 groupes. Par conséquent, cette solution est ignorée, et elle est remplacée par une autre solution qui consiste à calculer la meilleure partition d'une façon itérative, en partant en

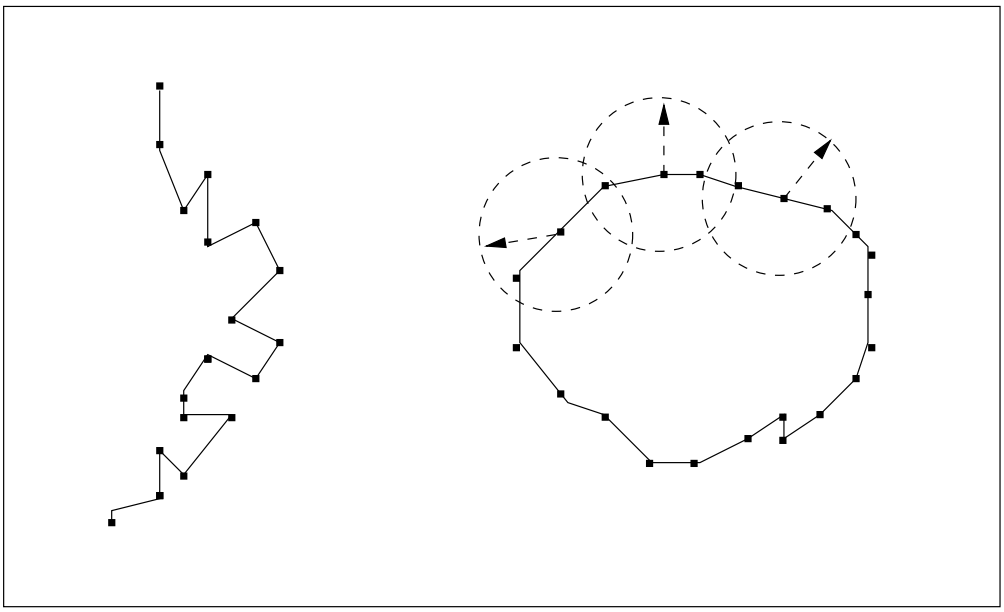


Figure 2.5 – Cette figure présente un exemple de clustering par l'approche de densité. Les points représentent les données et les cercles, le voisinage des données. Remarquons que ce type d'approche peut identifier les différentes formes de clusters. Chaque cluster est représenté par l'ensemble de données connectées.

général d'une partition initialisée aléatoirement.

Dans la suite de cette section, nous allons citer quelques méthodes non probabilistes. Certains d'entre elles réalisent le clustering *dur* et d'autres le *flou*.

Partitionnement *dur* (K-means) : la méthode *K-means* est l'une des techniques par partitionnement, qui est la plus connue et utilisée dans le domaine de clustering. Elle est très simple, et facile à appliquer. Le nom de cette méthode est inspirée de la façon de représenter les clusters, où chaque cluster est défini par un centroïde (moyenne pondérée des points dans le cluster), nommé aussi prototype. L'état initial de la méthode commence par le choix de K centroïdes aléatoirement dans l'espace de données. Ensuite, elle itère entre 2 étapes. Dans la première étape, chaque objet est affecté au prototype le plus proche (distance minimale). Puis dans la seconde étape, chaque prototype de chaque cluster est recalculé à partir des nouveaux objets associés à ce cluster. Ces deux étapes sont répétées jusqu'à ce qu'aucun changement ne soit produit, ou si la fonction de critère de convergence est réalisée. L'algorithme général de *K-means*, ainsi qu'un exemple d'application sont illustrés respectivement dans l'algorithme (2) et la figure (2.6).

L'algorithme de clustering *K-means* est simple à appliquer et efficace. Cependant il souffre de plusieurs limites :

– il favorise l'utilisation des données numériques puisque il représente ses clusters par des centres (obtenus par la moyenne des données associées aux clusters). Mais il peut être appliqué aussi sur

Entrées : k le nombre de clusters. L'ensemble de données $\{x_i\}$, avec $i = 1, \ldots, n$.

1 **Initialisation :** k prototypes $\{m_1, \ldots, m_k\}$, correspondent à k clusters $\{C_1, \ldots, C_k\}$, choisis aléatoirement

2 **répéter**

3 - assigner chaque objet x_i au cluster le plus proche, c'est-à-dire $x_i \in C_b$, si $||x_i - m_b|| < ||x_i - m_j||$ pour $i = 1, \ldots, n$ et $j = 1, \ldots, k$;

4 - Mettre à jour les prototypes, chacun d'eux à partir des objets qui lui sont associés. Si r objets sont attribués à C_b alors son nouveau prototype m_b est re-calculé comme suit :

$$m_b = \frac{\sum_r x_r}{r} \tag{2.1}$$

5 **jusqu'à** *ce qu'aucun changement ne se produisse ou si le critère de convergence est satisfait*;

Algorithme 2: Algorithme de *K-means*

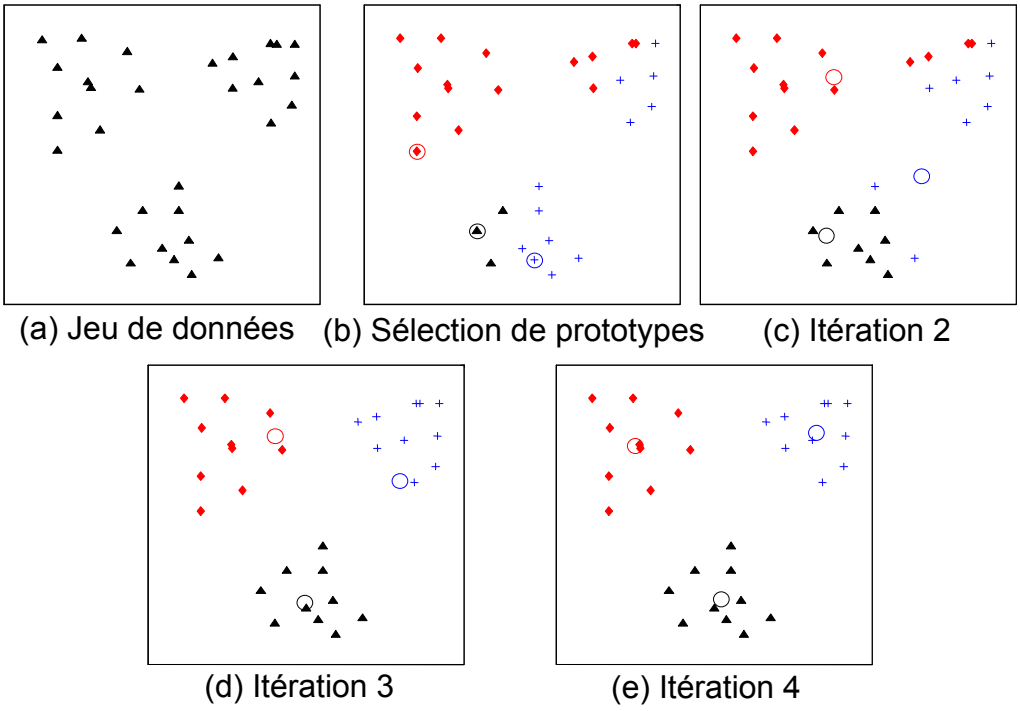


Figure 2.6 – Cette figure (extraite de [15]) illustre la progression de l'algorithme *K-means*. Dans la figure (a), les points noirs dénotent le jeu de données en deux dimensions. L'algorithme commence à choisir aléatoirement trois centres initiaux correspondant aux trois clusters. Puis, il associe chaque point au cluster le plus proche (distance minimale entre l'objet et le centre du classe)(figure b). L'algorithme itère (figures c et d) entre ces deux étapes jusqu'à la convergence (figure e).

les données catégoriques [62] ;

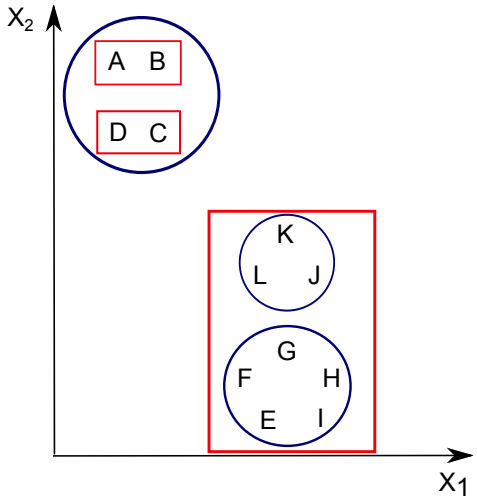


Figure 2.7 – Cette figure illustre bien le problème d'initialisation. Elle montre l'influence de la phase d'initialisation sur les résultats de clustering. Par exemple, si on lance l'algorithme *K-means* avec A, B, et C comme des prototypes initiaux de trois clusters, on obtient les trois clusters $C_1 = \{A, B\}$, $C_2 = \{C, D\}$, et $C_3 = \{E, F, G, H, I, J, K, L\}$. Ils sont représentés par les rectangles rouges dans la figure. Mais, si on choisit A, K, et E comme prototypes initiaux, on peut facilement avoir la meilleure partition $C_1 = \{A, B, C, D\}$, $C_2 = \{K, L, J\}$, et $C_3 = \{E, F, G, H, I\}$ qui est représentée par les cercles bleus.

- il est sensible à la présence de données atypiques. Ce type de données influence fortement les résultats (calcul de centres de composantes) ;
- il nécessite de prédéfinir le nombre de clusters a priori, et ses résultats dépendent directement de la phase d'initialisation (exemple 2.7) ;
- il a la tendance à produire seulement les clusters des formes sphériques ;
- il réalise un clustering *dur* en imposant à chaque donnée d'être associée à une seule composante.

Plusieurs extensions de l'algorithme *K-means* ont été proposées dans la littérature pour l'améliorer. Chacun d'entre elles a été étudiée pour résoudre un de ses problèmes basiques. Par exemple, les algorithmes suivants *ISODATA* [11], *K-medoids* [113], *CLARA* et *CLARANS* [84], ont été développés pour réduire les effets des données atypiques. Les algorithmes *CLARA* et *CLARANS* peuvent aussi réduire la complexité de calcul, surtout si le nombre de données est large. *K-modes* [62] fonctionne sur des données catégoriques .

Partitionnement *flou* : il existe aussi d'autres familles de clustering par partitionnement, qui réalisent le clustering *flou*. L'algorithme FCM (Fuzzy C-Means), est le plus connu parmi ce genre de méthodes. Il est similaire à l'algorithme *K-means*, mais il introduit le degré d'appartenance des objets dans le critère de convergence (erreur quadratique) à minimiser. Il nécessite aussi de calculer à chaque itération (étape d'estimation des paramètres) le degré d'appartenance de chaque donnée, afin de mettre à jour

les centres de clusters.

L'algorithme (FCM) a les mêmes limites (nombre de clusters, phase d'initialisation) que l'algorithme *K-means*. Plusieurs variantes de l'algorithme (FCM) et d'autres algorithmes de type clustering *flou* ont été aussi proposés pour améliorer les inconvénients de ce type d'algorithmes. Par exemple, les travaux de [25, 24] ont été développés pour faire face au problème de données atypiques. Ils ajoutent un cluster supplémentaire pour modéliser les données atypiques (voir la section 2.8). La famille des algorithmes de type *c-shells* a été étudiée pour détecter les différentes formes de clusters, en particulier les formes (lignes, anneaux, cercles, ellipses, rectangles, hyperboles) [33].

Comme dans notre approche, nous cherchons à calculer le modèle réduit à partir du modèle initial et d'une manière itérative (améliorer petit à petit les résultats jusqu'à ce qu'un critère d'arrêt soit vérifié), les approches par partitionnement sont pertinentes pour notre cas d'utilisation. Rappelons que nous avons motivé le choix d'utiliser des modèles probabilistes pour plusieurs raisons (représentation paramétrique, pertinente pour les applications distribuées). Le modèle initial (modèle probabiliste) est estimé à partir d'un jeu de données (dans le cas distribué plusieurs modèles sont générés à partir des données réparties). Pour réduire après ce modèle, nous avons choisi l'algorithme *K-means* connu par son efficacité et sa simplicité. Nous réalisons le même principe en calculant en première étape la distance entre les composantes du modèle initial et le modèle réduit, puis en mettant à jour les paramètres du modèle réduit selon les résultats obtenus dans la première étape. Les méthodes par partitionnement sont préférables pour notre objectif.

Nous avons présenté jusqu'à maintenant, la tâche de clustering d'un point de vue général. Nous avons montré pour cela les techniques principales étudiées pour réaliser le clustering. Nous avons aussi indiqué, que nous allions concentrer sur les méthodes probabilistes. Dans ce contexte, nous détaillons dans la section suivante, le principe de modèles de mélange probabiliste, ainsi que la façon d'estimer leurs paramètres par l'algorithme *EM* [28].

2.5 Modèle de mélange

Le modèle de mélange, par définition est un modèle statistique utilisé pour estimer paramétriquement la distribution (fonction de densité) de variables aléatoires en les modélisant comme une somme de plusieurs autres distributions simples.

Supposons l'existence d'une variable aléatoire $X = \{x_n | n \in 1, \ldots, N\}$, la densité de probabilité $f(x)$ du mélange peut s'exprimer comme la somme pondérée des autres densités $f_k(x)$ (composantes). Dans le cas où X est une variable discrète, elle prend la forme suivante :

$$f(x, \Theta) = \sum_{k=1}^{M} \pi_k f_k(x, \Theta_k) \tag{2.2}$$

où π_k représente la probabilité a priori de la composante k. Elle vérifie bien les conditions de probabilités tels que $\sum_k \pi_k = 1$, et $0 \leq \pi_k \leq 1$. Θ et Θ_k désignent respectivement les paramètres du modèle f et f_k.

Si X est une variable aléatoire continue, alors $f(x)$ s'écrit comme suit :

$$f(x, \Theta) = \int \pi_k f_k(x, \Theta_k) dx \tag{2.3}$$

Les densités de probabilités $f_k(x, \Theta_k)$ peuvent être une distribution (paramètre Θ_k) de la grande famille de distributions statistiques comme par exemple (Gaussien, Student). Le problème d'estimation d'un modèle de mélange, consiste à trouver une approximation appropriée de la densité $f(x)$ à partir d'un échantillon de n réalisations du vecteur aléatoire X. En d'autres termes, il sert à estimer les paramètres de la distribution de X en la modélisant comme une somme de plusieurs distributions f_k. Il s'agit alors de déterminer les paramètre Θ_k de chaque composante f_k. Dans la section suivante, nous allons montrer un exemple de modèle de mélange, le mélange gaussien.

L'objectif des modèles de mélange est de modéliser un échantillon de données. L'utilisation du modèle estimé peut être motivée ensuite pour réaliser plusieurs tâches :

- **prédiction :** le modèle estimé peut être utilisé pour prédire le cluster d'une nouvelle donnée (individu en probabilité). En effet, comme chaque cluster de données est estimé par un modèle de distribution probabiliste, alors le nouvel individu est affecté au cluster pour lequel sa vraisemblance est la plus élevée.

- **apprentissage distribué :** nous avons vu dans le chapitre (1), que l'estimation de données par un modèle de mélange peut être considérée comme une solution intéressante pour réaliser l'apprentissage distribué. Cela est du à sa représentation paramétrique qui entraîne une réduction en termes de coût de transmission et de calcul. Ainsi, il est très utile et pratique dans le cas où les données sont confidentielles.

- **affectation *flou* :** les résultats obtenus par l'estimation de modèles de mélange sont facilement interprétables, dus à la représentation probabiliste de données (chaque objet a un degré d'appartenance à chaque cluster). Il est possible d'affecter en *dur* les données en associant chaque objet au cluster présentant la plus forte probabilité.

2.6 Modèle du mélange gaussien

Afin de décrire le mélange gaussien, nous définissons tout d'abord la fonction de densité de la distribution gaussienne. Nous présentons ensuite la notion de mélange gaussien.

2.6.1 La distribution gaussienne

La distribution gaussienne, connue aussi sous le nom de la loi normale, est la distribution la plus connue parmi les distributions probabilistes. Elle a été largement utilisée pour modéliser la distribution des variables aléatoires continues. Dans le cas d'une variable aléatoire simple X, la fonction de densité de la loi gaussienne peut s'écrire sous la forme suivante :

$$\mathcal{N}(x|\mu_k, \sigma_k^2) = \frac{1}{(2\pi)^{d/2}\sigma^{1/2}} e^{(-1/2\sigma_k^2)(x-\mu_k)^2)} \tag{2.4}$$

où μ_k est la moyenne, et σ_k^2 est la variance. Si la variable X est d-dimensionnelle, alors la fonction de densité de la loi gaussienne multivariée prend la forme suivante :

$$\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{\frac{-1}{2}(x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k)} \tag{2.5}$$

où μ_k est un vecteur de d-dimension, Σ_k est la matrice de covariance de dimension $d * d$, et $|\Sigma_k|$ désigne le déterminant de la matrice Σ_k.

2.6.2 Mélange gaussien

Le mélange gaussien a été utilisé dans plusieurs domaines comme la reconnaissance de formes, l'apprentissage statistique [45, 20]. Il peut modéliser n'importe quel jeu de données numériques, mais avec une précision arbitraire.

Le modèle de mélange gaussien est une combinaison linéaire de plusieurs composantes gaussiennes. Il est particulièrement utilisé dans le cas où les données en études ne peuvent pas être modélisées par une simple gaussienne. En d'autres termes, si la structure de données est composée naturellement par plusieurs groupes, il faut les représenter par un modèle de mélange gaussien plutôt qu'une simple distribution gaussienne.

Pour le formaliser, il suffit juste de remplacer chaque $f(x, \Theta_k)$ par la fonction de densité de la loi gaussienne. La fonction de densité de la loi gaussienne s'écrit alors comme suit :

$$f(x|\Theta) = \sum_{k=1}^{M} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k). \tag{2.6}$$

où $\mathcal{N}(x|\mu_k, \Sigma_k)$ est la composante k dans le mélange définie comme dans l'équation (2.5).

Dans ce contexte, on définit l'ensemble des poids par $\pi = \{\pi_k\}$, l'ensemble des moyennes par $\mu = \{\mu_k\}$, et celle des covariances par $\Sigma = \{\Sigma_k\}$. On définit également $\Theta = \{\Theta_k\}$ et $\Theta_k = \{\pi_k, \mu_k, \Sigma_k\}$. Un exemple d'un mélange gaussien composé de 2 composantes gaussiennes est illustré dans la figure 2.8.

Après avoir défini le mélange gaussien, nous allons voir dans la section suivante plusieurs utilisations importantes de ce mélange dans plusieurs domaines.

2.6.3 Utilisations

Les mélanges de gaussiennes sont très utilisés dans plusieurs domaines d'applications (reconnaissance de formes, recherche d'informations). En particulier, ils sont considérés comme un outil important, pour modéliser et traiter les données multimédia (images, audio, vidéo). Dans le reste de cette section,

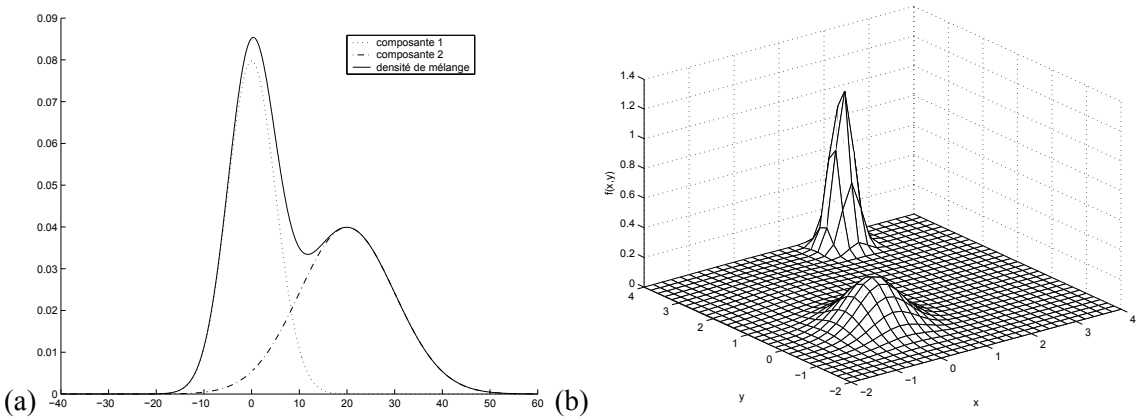


Figure 2.8 – Cette figure illustre un exemple d'un mélange gaussien de 2 composantes (extraite de [94]). Les courbes pointillées dans la figure (a) représentent les composantes gaussiennes et la courbe continue illustre la densité de mélange de ces deux gaussiennes. La figure (b) montre la densité $(f(x, y))$ de composantes gaussiennes en 3 dimensions.

nous citons quelques unes de ses utilisations.

Par exemple dans le domaine de recherche d'images par le contenu, il existe de nombreuses manières de caractériser les ressemblances entre les images (texture, couleur, formes, combinaison de plusieurs de ces caractéristiques [20, 45]). Une solution alternative pour représenter les images, est d'appliquer une technique probabiliste qui se base sur les modèles de mélange gaussien. Par exemple, dans [45], les pixels dans chaque image sont regroupés selon leurs intensités et leurs position, ou (intensité, position, texture) comme dans [20], dans des groupes homogènes (réalisés par un algorithme d'estimation comme *EM* [28]). Par conséquent, chaque image est représentée par un modèle de mélange gaussien. Cette méthode est motivée par le fait, qu'il est plus facile de calculer la similarité entre deux images, en utilisant cette représentation plutôt qu'une autre (par exemple les pixels). Donc, il sera plus rapide de rechercher les images les plus similaires à une requête d'utilisateur parmi une collection d'images en se basant sur la représentation probabiliste.

Les modèles de mélange gaussien sont utilisés aussi dans le domaine de reconnaissance audio. Ils sont appliqués par exemple, pour modéliser les locuteurs comme dans [98].

Nous avons présenté dans cette section plusieurs utilisations du modèle de mélange gaussien. Ensuite, nous expliquons comment les paramètres du modèle de mélange gaussien peuvent être estimés, en utilisant le technique la plus connue dans ce domaine : l'algorithme *EM*.

2.7 Estimation de paramètres des modèles de mélange

Plusieurs méthodes d'estimation sont déjà étudiées dans la littérature pour estimer les paramètres de la distribution de probabilités d'un échantillon donné [99], dont la plus connue est la méthode de maximum de vraisemblance [15].

Dans cette section, nous nous concentrons dans un premier temps, sur l'estimation de modèle de mélange par la méthode de maximum de vraisemblance. Puis nous détaillons dans un deuxième temps, l'algorithme le plus utilisé pour estimer le modèle de mélange, l'algorithme de *EM*.

2.7.1 Maximum de vraisemblance

L'idée fondamentale de l'estimation par maximum de vraisemblance est, comme son nom l'indique, de trouver un ensemble d'estimations des paramètres, pour que la vraisemblance de l'échantillon utilisé, soit maximum.

Étant donné la même hypothèse, que les donnés X= $\{x_1 \ldots, x_n\}$ sont des réalisations indépendantes d'un vecteur aléatoire X, la loi de vraisemblance des données relativement au modèle de paramètre Θ s'écrit :

$$L(\Theta) = \prod_{i=1}^{n} f(x_i|\Theta) \tag{2.7}$$

Il est plus facile de maximiser la log-vraisemblance au lieu de la fonction de vraisemblance elle-même. Si Θ maximise $ln\,(L(\Theta))$, alors il maximise également $L(\Theta)$. Cela est du à la monotonie de la fonction logarithme. La fonction log-vraisemblance s'écrit :

$$ln\,(L(\Theta)) = \sum_{i=1}^{n} ln\,(f(x_i|\Theta)) \tag{2.8}$$

Le problème d'estimation par la méthode de maximum de vraisemblance, revient donc à trouver les racines de l'équation :

$$\frac{\partial ln\,(L(\Theta))}{\Theta} = 0 \tag{2.9}$$

En général, la maximisation de la fonction de vraisemblance ne possède pas de solution analytique. C'est pour cela qu'il est nécessaire de recourir à des méthodes itératives. Dans la partie suivante, nous allons expliquer en détail, la façon de trouver le maximum de vraisemblance en utilisant l'algorithme *EM*.

2.7.2 Algorithme *EM* standard

L'algorithme (EM) Expectation-Maximization est la méthode la plus employée pour estimer les paramètres d'un modèle de mélange. C'est une technique itérative, permettant de maximiser la vraisemblance des paramètres de modèles probabilistes, en présence de variables latentes (non observables). Il a été principalement utilisé pour classer des données, qui sont supposées générées à partir d'un modèle de mélange probabiliste (par exemple le mélange gaussien). Il s'agit donc d'un algorithme itératif, qui

Entrées : $\Theta^{(0)}$ une valeur initiale des paramètres du modèle, X un ensemble d'observations, ϵ un seuil pour la convergence de l'algorithme
Sorties : le modèle Θ maximisant la vraisemblance

1 t$\rightarrow 0$;
2 **répéter**
3 **(E-step)** Calcul de l'espérance conditionnelle de la fonction de vraisemblance $\mathrm{Q}(\Theta|\Theta^{(t)}) = \mathrm{E}[\mathrm{L}(\Theta)|X, \Theta^{(t)}]$;
4 **(M-step)**Maximisation de $\mathrm{Q}(\Theta|\Theta^{(t)})$ $\Theta^{(t+1)} = \underset{\Theta}{\operatorname{argmax}}\, Q(\Theta, \Theta^{(t)})$;
5 $t = t + 1$;
6 **jusqu'à** $\left|\mathrm{Q}(\Theta|\Theta^{(t+1)}) - \mathrm{Q}(\Theta|\Theta^{t})\right| < \epsilon$;

Algorithme 3: l'algorithme EM stantard

consiste à estimer en alternance, les probabilités a posteriori des variables latentes (étape E) et les paramètres du modèle (étape M) (voir l'algorithme (3) pour plus de détails).

Supposons qu'à chaque individu x, une variable latente $z \in \mathbb{R}^M$ est associée. La variable latente est de type "vecteur canonique", c'est-à-dire $z \in \{e_1, \ldots, e_M\}$, avec e_k le kème vecteur canonique de IR^M et z_k la kème dimension du vecteur z. $z_k = 1$ ou $z_k = 0$ selon que, x appartient ou n'appartient pas à la kème composante.

La fonction log-vraisemblance de X en introduisant les variables latentes Z prend la forme suivante :

$$ln\left(f(X|\Theta)\right) = ln\left\{\sum_{z} f(X, Z|\Theta)\right\}. \tag{2.10}$$

Cependant, il est compliqué de maximiser directement la log-vraisemblance de $f(X|\Theta)$, et l'ensemble X de données est supposé incomplet. Pour la suite, notons par l'ensemble {X,Z} les données complètes, et remplaçons la fonction log-vraisemblance de $f(X|\Theta)$ par celle de $f(X, Z|\Theta)$, qui est plus facile à manipuler. Par exemple, la fonction log-vraisemblance de données complètes, en terme de mélange gaussien s'écrit :

$$ln\left(f(X, Z|\Theta)\right) = \sum_{n=1}^{N}\sum_{k=1}^{M} z_{nk}\{ln\pi_k + ln\left(\mathcal{N}(x_n|\mu_k, \Sigma_k)\right)\}. \tag{2.11}$$

où z_{nk} dénotes la $keme$ composante de z_n.

Notons par $Q\left(\Theta, \Theta^{old}\right)$, l'espérance mathématique de l'équation (2.11), où Θ^{old} représente l'ensemble des paramètres courants.

Pour estimer par exemple les paramètres d'un mélange gaussien, l'algorithme *EM* itère entre les deux étapes suivantes jusqu'à la convergence : l'étape d'Estimation et celle de Maximisation.

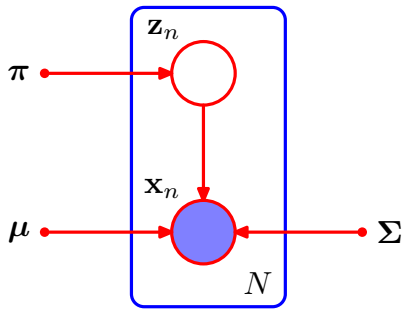


Figure 2.9 – Représentation graphique d'un mélange gaussien en présence des variables latentes, par un graphe orienté acyclique. Le cercle vide à l'intérieur du rectangle indique une distribution inconnue (la variable latente dans ce cas), et le cercle plein indique que cette variable est observée. La flèche orientée représente une dépendance conditionnelle. N est la taille de la population, et π, μ, et Σ sont les paramètres ajustables du modèle.

– **Étape Estimation :** dans cette étape, les paramètres courants sont utilisés pour calculer l'espérance de z_{nk}, noté par $\gamma(z_{nk})$, comme suit :

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{M} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}. \tag{2.12}$$

– **Étape Maximisation :** après avoir calculé les espérances de z_{nk} dans l'étape E, la quantité $Q\left(\Theta, \Theta^{old}\right)$ doit être maximisée par rapport aux différents paramètres du modèle. Ce n'est autre que l'étape M, qui consiste à mettre à jour les paramètres de modèle selon l'équation suivante :

$$\Theta^{new} = \underset{\Theta}{\operatorname{argmax}}\, Q(\Theta, \Theta^{old}). \tag{2.13}$$

Les nouveaux paramètres estimés Θ^{new} sont donnés par :

– les proportions :

$$\pi_k^{new} = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_{nk}), \tag{2.14}$$

– les moyennes :

$$\mu_k^{new} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) x_n}{\sum_{n=1}^{N} \gamma(z_{nk})}, \tag{2.15}$$

– les matrices de covariances :

$$\Sigma_k^{new} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^t}{\sum_{n=1}^{N} \gamma(z_{nk})}. \tag{2.16}$$

Entrées : $\Theta^{(0)}$ une valeur initiale des paramètres du modèle, X un ensemble d'observations, M le nombre de composantes ou classes
Sorties : une estimation $\tilde{\Theta}$ des paramètres du modèle

k$\rightarrow 0$;
Initialisation du modèle $\Theta^{(0)}$;
répéter
  (Étape -E)(Estimation)
  pour $n = 1$ *à N* **faire**
    pour $k = 1$ *à M* **faire**

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{M} \pi_j \mathcal{N}(x_n|\mu_j,\Sigma_j)} ;$$

    fin
  fin
  (Étape-M)(Maximisation)
  pour $k = 1$ *à M* **faire**

$$\pi_k^{new} = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_{nk});$$

$$\mu_k^{new} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) x_n}{\sum_{n=1}^{N} \gamma(z_{nk})};$$

$$\Sigma_k^{new} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^t}{\sum_{n=1}^{N} \gamma(z_{nk})};$$

  fin
  $k = k + 1$;
jusqu'à *Convergence*;

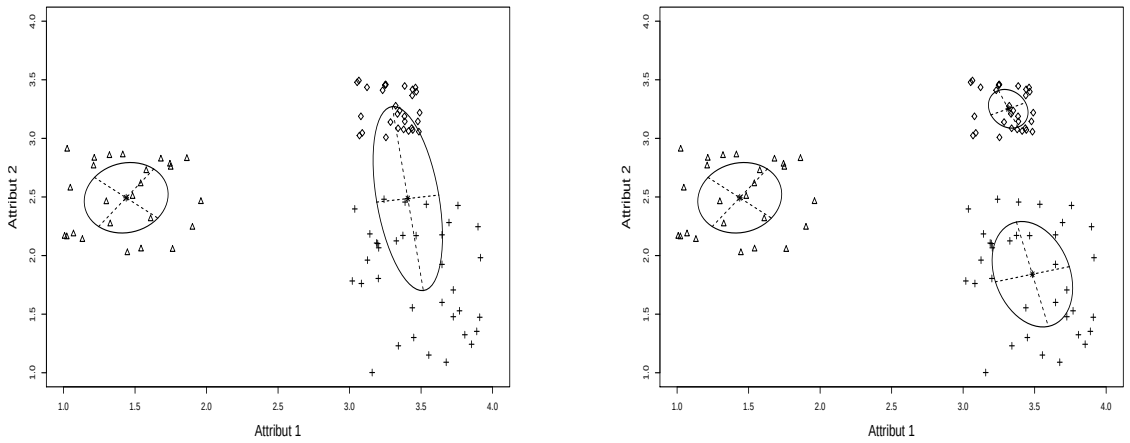
Algorithme 4: l'algorithme *EM* pour le mélange gaussien

La convergence de l'algorithme est réalisée dans l'un de ces deux cas : soit la différence de la fonction de vraisemblance entre deux étapes consécutives est négligeable, soit les nouveaux paramètres estimés ne changent pas par rapport à l'étape précédente.

L'algorithme *EM* converge sûrement vers un optimum local de vraisemblance. Cette convergence dépend directement de la phase d'initialisation de l'algorithme. Une mauvaise initialisation pourrait conduire à un mauvais modèle, alors qu'une bonne initialisation entraînera plutôt une convergence vers un optimum global de vraisemblance.

L'algorithme (4) décrit les différentes étapes de l'algorithme *EM* appliqué sur un mélange gaussien.

L'algorithme standard *EM* a plusieurs limites majeures, et ses résultats dépendent directement du choix des paramètres : le nombre de composantes et les paramètres initiaux du modèle. Pour cette raison,



(a) (b)

Figure 2.10 – Cette figure montre bien le problème de la détermination du nombre de clusters optimal, sur un jeu de données composé de trois clusters identifiables visuellement. Les figures (a) et (b), représentent respectivement un exemple d'application de l'algorithme de clustering *EM* avec deux et trois clusters. Nous pouvons conclure facilement, que les résultats de clustering dépendent directement du choix du nombre de clusters (fixé a priori).

plusieurs extensions et variantes de cet algorithme ont été développées pour l'améliorer et résoudre ces problèmes [21, 111, 130, 83].

Nous avons présenté dans cette section l'estimation d'un modèle de mélange gaussien par l'algorithme *EM*. Nous nous concentrons ensuite sur le problème des données atypiques dans les méthodes de clustering.

2.8 Problème de la détection des données atypiques

La détection des données atypiques se réfère au problème de la découverte d'objets qui s'écartent significativement de la majorité des objets. Elle peut être décrite comme suit : étant donné un ensemble n d'objets et k le nombre prévu d'objets atypiques, le but est de trouver les k objets qui sont considérablement dissemblables, exceptionnels, ou incompatibles avec les objets restants. La détection des données atypiques a été largement étudiée dans plusieurs domaines comme les statistiques, la fouille de données, l'apprentissage statistique, la théorie de l'information, etc [128, 22]. Aussi, elle a été appliquée dans plusieurs domaines d'application, comme la détection de fraude, la détection d'intrusion, l'analyse de performances [128, 22]. Les données atypiques peuvent être par exemple d'objets (individus) qui ne font pas partie de la population en étude, ou bien une erreur de saisie ou de mesure.

Le but de cette section est de présenter les techniques principales existantes pour la détection des données atypiques. Il existe des approches qui s'adressent seulement à ce genre de problème (c'est-à-dire que leur objectif est d'identifier seulement les données atypiques) et d'autres intégrant la détection des données atypiques comme une étape intermédiaire pour réaliser d'autres tâches (comme le clustering

dans notre cas). Nous présentons en général les techniques les plus courantes [61, 128, 22] en décrivant leurs principes et en mettant en valeur leurs propriétés principales. Nous allons étudier en particulier les méthodes basées sur le clustering, les méthodes basées sur les plus proches voisins, les méthodes tronquées, les méthodes basées sur les modèles, les méthodes basées sur la théorie de l'information et les méthodes basées sur la décomposition spectrale.

2.8.1 Méthodes basées sur le clustering

Ces méthodes se basent directement sur les approches de clustering pour identifier les données atypiques. Les données normales (typiques) appartiennent aux clusters denses tandis que les données atypiques sont celles n'appartenant à aucun cluster ou bien appartenant à un cluster qui contient un nombre très petit de données [95, 22, 61, 128, 129].

Plusieurs algorithmes de clustering cherchent à trouver les données atypiques en utilisant les résultats de clustering. Mais leur objectif principal est de partitionner les données et non pas d'identifier ce type de données. Par exemple la méthode *CURE* [127] qui est une technique hiérarchique ascendante, a été développée à la base pour réaliser des clusterings. Elle élimine les clusters contenant un nombre très petit de données. La phase d'élimination est faite seulement à partir du moment où le nombre de clusters fusionnés devient inférieur au nombre de clusters initiaux selon un seuil précis (paramètre d'utilisateur).

La méthode *CHAMELEON* consiste à effectuer aussi un clustering hiérarchique ascendant [72]. Elle fusionne à chaque itération les deux clusters pour lesquels la similarité calculée entre eux, est supérieure à un seuil défini par l'utilisateur. Si à une itération donnée, la condition de fusionnement n'est pas vérifiée (valeur de similarité entre tous les clusters inférieure au seuil) alors les clusters contenant un nombre très petit de données sont déclarés comme atypiques.

La méthode étudiée dans [58] se base aussi sur les résultats de clustering mais en calculant de plus, un facteur d'atypicité (ou facteur d'aberration) de chaque donnée. Ce facteur est obtenu en fonction de la taille du cluster qui contient la donnée et sa distance au centre du cluster le plus proche. Les données de plus grande valeur (pourcentage défini par l'utilisateur) sont déclarées comme atypiques.

Notons qu'il existe des méthodes qui se basent sur la technique de classification supervisée (phase d'apprentissage et phase de test) pour identifier les données atypiques. Dans ces méthodes, deux classes sont définies, une pour les données normales et l'autre pour les données atypiques. Durant la phase de test, un nouveau point est associé à l'une de ces deux classes apprises. Dans le domaine semi-supervisé, une seule classe de données normales est définie. Si un nouveau point se trouve en dehors de cette classe, alors il est déclaré comme un point atypique. Plusieurs types d'approches basées sur la classification existent : les approches basées sur la machine à vecteurs de support et les approches basées sur les réseaux bayésiens et les réseaux de neurones [22, 61].

Nous avons vu que la première étape de méthodes basées sur les approches de clustering (ou classification) implique une phase de clustering (ou classification). La détection des données atypiques dépend donc directement des résultats de cette étape, nécessitant de choisir des paramètres convenables pour réaliser un bon clustering (nombre correct de clusters, phase d'initialisation) ou de trouver un bon clas-

sifieur (c'est-à-dire qu'il peut séparer correctement les données normales et atypiques).

2.8.2 Méthodes basées sur les plus proches voisins

Ces approches sont les plus utilisées pour la détection des données atypiques [22, 61, 128, 129]. Elles analysent chaque point par rapport à son voisinage (à l'inverse par exemple, des méthodes basées sur le clustering qui nécessitent une vision globale de toutes les données). Elles peuvent être divisées en deux catégories selon lesquelles les données sont analysées par rapport à leurs voisins. La première catégorie mesure la distance entre chaque point et ses plus proches voisins, tandis que la deuxième catégorie compare la densité de chaque point avec la densité de ses voisins. Dans la suite, nous montrons comment les deux types d'approches peuvent identifier les données atypiques.

Méthodes basées sur les distances : Ces approches se basent donc sur la distance entre les points et leurs proches voisins. Pour détecter les données atypiques dans ces approches, plusieurs tests peuvent être appliqués. Par exemple, si la distance d'un point par rapport à son plus proche voisin (ou a $keme$ plus proche voisin) dépasse un seuil d (défini par l'utilisateur) alors il est déclaré comme atypique [31, 76]. Les auteurs de [97] calculent un score pour chaque donnée (sa distance au $keme$ plus proche voisin). Ils déclarent les données qui ont les scores les plus élevés comme atypiques. Les travaux de [31, 5] utilisent plutôt la somme des distances entre chaque donnée p et ses k plus proches voisins comme un score d'atypicité de p.

Méthodes basées sur la densité des points : Ce type d'approches évalue aussi chaque point par rapport à ses plus proches voisins. Pour identifier les données atypiques, les méthodes basées sur la densité des points utilisent la définition suivante : un point p est un atypique s'il ne contient pas un nombre minimal (k) de points qui ont une distance inférieure à d (c'est-à-dire dans un cercle de voisinage de rayon d) [76, 131]. Elles calculent donc la densité des régions de données et elles déclarent les données qui se situent dans des régions faibles comme atypiques.

Rappelons que les méthodes de clustering basées sur la densité de points fournissent des protections naturelles contre les données atypiques (voir la section 2.4.3). Elles séparent les régions denses (clusters normaux) de celles de densités faibles (données atypiques) [60, 6, 122, 105]. Les clusters (ou les régions denses) sont les ensembles de données connectées contenant au moins un nombre précis k de points dans leur voisinage (cercle de rayon d). Les données atypiques sont celles qui n'appartiennent à aucun cluster, après la terminaison de processus de clustering (c'est-à-dire celles qui se trouvent dans des régions faibles).

Les méthodes citées ci-dessus ont été développées à la base pour réaliser un clustering. Cependant il existe d'autres approches basées sur la densité de points qui ont pour but de détecter seulement les données atypiques [110, 17]. Ces méthodes sont étudiées pour faire face au problème de la détection des données atypiques, dans les données de densités différentes (difficile de les identifier avec les méthodes basées sur la distance). Elles prennent en compte les densités locales des points (nombre de données dans leur voisinage), et elles évaluent pour chaque point sa densité par rapport à celles de ses voisins. Un exemple simple qui montre bien l'efficacité des approches de densités, par rapport à celles basées

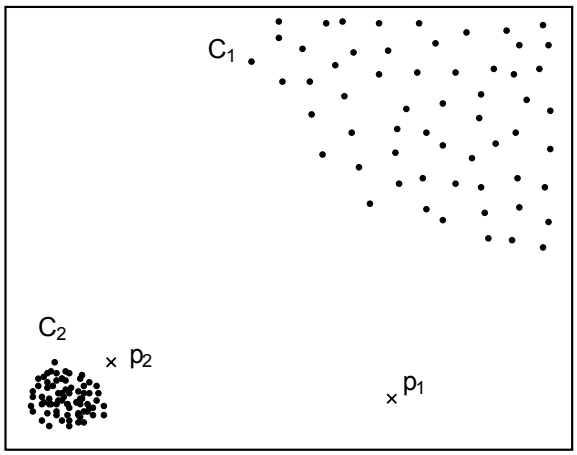


Figure 2.11 – Cette figure a été extraite de [17]. Elle contient deux clusters (C_1 et C_2) avec des densités différentes et deux points atypiques (p_1 et p_2). Le but de cette figure est de montrer qu'une méthode basée sur la densité peut identifier le point p_2 contrairement aux méthodes basées seulement sur la distance entre les points et leurs plus proches voisins.

sur la distance est illustré dans la figure (2.11). Cette figure contient deux clusters de points (C_1 de densité faible et C_2 de densité élevée) et deux points atypiques p_1 et p_2. Dans les méthodes basées sur la distance, le point p_2 est considéré typique (normal). Ceci est due à la distance de p_2 à son plus proche voisin dans C_2 qui est inférieur à celle entre chaque point dans C_1 et son plus proche voisin. Cependant les méthodes de densités sont capables d'identifier le point atypique p_2 en tenant compte de la densité autour de chaque point.

Les approches basées sur les plus proches voisins sont simples et faciles à appliquer, mais les résultats de ces méthodes dépendent directement des paramètres définis par l'utilisateur (le nombre de voisins k, le rayon de voisinage d). Par exemple, pour augmenter le nombre des données atypiques il faut soit initialiser le paramètre k par une valeur grande et fixer d, soit diminuer d et fixer k. Dans certains cas, il n'est pas facile de trouver une valeur pertinente de ces paramètres comme par exemple dans le cas de paramètre d (si les données sont multidimensionnelles ou si les clusters sont de densités très différentes).

2.8.3 Méthodes tronquées

Les méthodes tronquées cherchent à éliminer le pourcentage de données les plus éloignées. Un exemple simple consiste à calculer la moyenne tronquée d'un échantillon de données univariées après la suppression d'un pourcentage de $\alpha/2$ des données les plus larges et un pourcentage de $\alpha/2$ des données les plus petites (α est un paramètre d'utilisateur). Dans le cas des données multivariées, deux méthodes DCM (déterminant de covariance minimale) [100, 79] et VEM (volume d'ellipsoïde minimale) [39] sont utilisées comme des procédures très robustes pour estimer le modèle d'un ensemble de données (location, dispersion). Elle peuvent réaliser des bonnes estimations, même en présence d'une grande quantité de données atypiques (jusqu'au un pourcentage de 50% de données traitées) [100, 79].

Les estimateurs DCM de localisation et de dispersion sont la moyenne et la matrice de covariance calculées sur l'échantillon de $h = n(1 - \alpha)$ (paramètre utilisateur) points parmi n (nombre de données) qui minimise le déterminant de la matrice de covariance correspondante. L'idée principale de la technique DCM consiste à trouver une proportion de h de données (étape de concentration) ayant les distances de *Mahalanobis* minimales avec la moyenne (initialisée aléatoirement à la première itération). Puis la moyenne et la matrice de covariance sont mises à jour itérativement à partir de données obtenues dans l'étape précédente. L'algorithme *FAST-MCD* implémentant cette technique peut être résumé par les étapes suivantes :

- il sélectionne plusieurs échantillons de h points ;
- il calcule pour chacun d'eux la moyenne et la covariance ;
- il met à jour les paramètres (moyenne, covariance) de chaque échantillon à partir de h points ayant une distance minimale avec la moyenne et la covariance de l'itération précédente ;

Ces deux dernières étapes sont répétées jusqu'à obtenir le meilleur échantillon ayant un déterminant de covariance minimale. La technique DCM a été très utilisée avec plusieurs méthodes de clustering robustes [44, 41].

Par exemple la méthode *K-means tronqué* a été proposée pour rendre l'algorithme de *K-means* robuste face aux données atypiques [44, 41]. Elle utilise la technique DCM en calculant pour chaque cluster les h points les plus proches à son centre. Puis elle met à jour le centre de chaque cluster uniquement à partir de points obtenus dans l'étape précédente (h points). La fonction à optimiser est la même que celle de l'algorithme de *K-means* (erreur quadratique), mais en utilisant seulement les h points de chaque cluster et non pas toutes les données.

L'algorithme *TCLUST* intègre la technique de DCM dans l'algorithme *EM* pour estimer le modèle de mélange [44, 41]. Il adapte la technique de DCM en introduisant les étapes Estimation et Maximisation de l'algorithme *EM*. Pour chaque cluster, l'ensemble de h points qui ont les probabilités a posteriori (étape E de l'algorithme *EM*) maximales est gardé. Puis, les paramètres de chaque composante de modèle (étape M) sont mis à jour à partir de points obtenus dans la première étape. La fonction à optimiser est celle de l'algorithme *EM* [28].

Les techniques tronquées cherchent à éliminer la proportion des données (paramètre h) les plus éloignées (supposées atypiques). Le nombre de données atypiques peut être donc contrôlé par la valeur de h. h est considéré comme le seul paramètre critique des méthodes tronquées (si sa valeur est petite ceci peut entraîner de ne pas éliminer toutes les données atypiques et si sa valeur est grande cela peut conduire à ne pas choisir des données normales qui peuvent être significatives).

Dans cette thèse nous nous basons sur ce type de techniques en éliminant des modèles non fiables (représentant les données) plutôt que les données. Nous allons voir cela en détail dans la section (5.3) du chapitre (5).

2.8.4 Méthodes basées sur les modèles

Ces méthodes cherchent à représenter les données atypiques par un modèle. On peut distinguer deux types de modèles : prototypes ou probabilistes. Nous montrons dans la suite plusieurs techniques pour représenter les données atypiques par ce genre de modèles.

Représentation des données atypiques par des modèles de prototypes : Ces approches consistent à modéliser les données atypiques en utilisant les prototypes comme des représentants de clusters. Il existe deux manières pour ignorer l'influence des données atypiques : soit en définissant une représentation robuste des clusters [113], soit en modélisant les données atypiques par un cluster supplémentaire [24].

La méthode *K-medoids* représente chaque cluster par sa donnée la plus représentative (*medoid*) [113]. Cette définition permet donc d'obtenir un modèle peu sensible aux données atypiques. Cette approche a été développée pour robustifier l'algorithme *K-means*. La fonction à optimiser (similaire à celle utilisée pour l'algorithme *K-means*) consiste à minimiser la somme des distances des données à leur *medoid*. Cette approche (similaire à *K-means*) dépend de paramètres arbitraires (nombre de clusters, phase d'initialisation). Elle réalise un clustering robuste plutôt que de modéliser les données atypiques ou les identifier. Mais une technique simple peut être appliquée pour déterminer les données atypiques : en déclarant les données qui ont les distances les plus éloignées avec le *medoid* de leur cluster comme atypique.

Les travaux de [24] supposent l'existence d'un cluster supplémentaire pour modéliser les données atypiques. Ils ont considéré aussi que ce cluster est défini par un prototype qui a une distance constante δ avec tous les points en étude. Le but de leur technique est de réaliser un clustering *dur* d'une manière robuste. Chaque point (typique ou atypique) a un degré d'appartenance dans les clusters normaux (typiques) et dans le cluster de données atypiques. La fonction de convergence à optimiser s'écrit comme suivante :

$$\mathcal{J} = \sum_{i=1}^{K}\sum_{j=1}^{n}(u_{ij})^2 d_{ij}^2 + \sum_{j=1}^{n}\delta^2(1-\sum_{i=1}^{K}u_{ij})^m \tag{2.17}$$

où u_{ij} désigne le degré d'appartenance de point j au cluster i et et d_{ij} est la distance de point j au centre de cluster i.

La constante δ signifie que tous les points (même les données atypiques) ont la même probabilité a priori d'appartenir au cluster de données atypiques. Elle est initialisée en fonction de la moyenne des distances d_{ij}^2 (voir [24] pour plus de détails).

Cette technique peut être appliquée pour rendre une grande variété des approches basées sur les prototypes plus robustes [26]. Cependant, les résultats dépendent de la valeur de δ (il n'est pas facile de trouver une valeur pertinente de ce paramètre).

Représentation des données atypiques par des distributions statistiques : Ce type de méthodes est similaire aux méthodes basées sur les prototypes, mais suppose que les données sont générées à partir d'une distribution probabiliste connue. Ces approches cherchent aussi à modéliser les données atypiques soit en utilisant une distribution robuste comme la distribution de *Student* soit en ajoutant d'autres distri-

butions pour les représenter [38, 93, 85].

Les travaux de [38] ont proposé une technique robuste pour estimer un modèle de mélange gaussien en présence de données atypiques. Ils ont considéré que chaque donnée x_i est générée par une composante gaussienne avec une probabilité $(1-\epsilon)$ plus une distribution h_i des données atypiques avec une probabilité ϵ (ϵ dépend de la quantité de données atypiques prévue dans l'ensemble de données). La fonction de densité du point x_i s'écrit comme suit :

$$p(x_j|\theta) = (1-\epsilon)\sum_{i=1}^{K}\mathcal{N}(x_j, \mu_i, \Sigma_i) + \epsilon\sum_{i=1}^{K}h(x_j) \tag{2.18}$$

Pour identifier les données atypiques, cette méthode a considéré que la fonction de vraisemblance d'un point normal (typique) généré par une composante gaussienne est supérieure à la vraisemblance de n'importe quel point atypique, généré aussi par la même gaussienne. Elle a supposé aussi que la distribution h_i est identique pour tous les points traités et égale à une constante δ_i. Cette méthode a été testée seulement pour des données des dimensions faibles.

Les auteurs de [93] ont utilisé les mélanges de *Student* pour réduire l'effet de données atypiques. La distribution de Student est connue dans le domaine de statistique pour sa robustesse. Elle est caractérisée par un paramètre nommé le degré de liberté qui permet de régler la robustesse de modèle. Les données qui sont atypiques pour une composante donnée ont eu un poids réduit dans le calcul de ces paramètres. Leur technique consiste à appliquer l'algorithme *EM* pour estimer les paramètres de modèle de *Student*. Dans l'étape *Estimation* la probabilité a posteriori de chaque donnée est calculée. Puis les paramètres de modèle sont mis à jour dans l'étape (Maximisation). La fonction de vraisemblance adaptée au mélange de *Student* a été utilisée comme un critère de convergence. Des extensions de l'algorithme *EM Student* ont été étudiées dans [78, 93] pour accélérer sa convergence.

Notons que nous avons proposé dans cette thèse une technique robuste de réduction de modèle de mélange en utilisant des mélanges de *Student*.

Les approches basées sur les modèles de mélange ne nécessitent pas de paramètres définis par les utilisateurs. Les paramètres de la distribution sont calculés directement à partir de données originales (seule une initialisation des paramètres peut être requise). Cependant elles imposent que la distribution a priori de données soit connue.

2.8.5 Méthodes basées sur la théorie de l'information

Les méthodes basées sur la théorie de l'information consistent à analyser le contenu d'informations dans un ensemble de données. À ce titre, il existe plusieurs critères pour mesurer la quantité d'information comme l'entropie, l'entropie relative. L'idée générale de ces approches est d'évaluer l'influence de chaque point sur le contenu d'informations de données et de déclarer le point qui l'affecte comme atypique.

Par exemple les travaux de [89, 22] cherchent à calculer d'abord pour chaque point p l'ensemble de ses voisins V (points les plus proches de p selon la distance euclidienne). Puis ils appliquent la divergence de *Kullback-Leibler* pour mesurer la similarité entre V et $V \cup \{p\}$. Le point qui cause la plus grande

divergence est considéré comme atypique. Ce point est éliminé et l'opération est répétée en supprimant un point à chaque itération jusqu'à trouver toutes les données atypiques (un critère de convergence est vérifié).

Ce type d'approches cherchent donc à maximiser la quantité d'informations dans l'ensemble de données. Cependant il est difficile de trouver un seuil pour les critères d'information à partir duquel un point est déclaré ou pas comme atypique.

2.8.6 Méthodes basées sur la décomposition spectrale

Les approches basées sur la décomposition spectrale utilisent la méthode d'analyse de composantes principales pour trouver les données normales. L'objectif principal de ces méthodes est de trouver parmi toutes les composantes, qui représentent les différentes dimensions, celles qui décrivent mieux le comportement des données. Un point est considéré atypique s'il ne correspond pas à cette structure de composantes [128].

Ce type de méthodes paraît convenable pour détecter les données atypiques des données multidimensionnelles. Cependant, la recherche des composantes principales représentant le mieux l'ensemble de données est très coûteuse.

Nous avons présenté dans cette section les techniques les plus utilisées pour détecter les données atypiques (en particulier celles appliquées dans les méthodes de clustering). Nous allons montrer dans les chapitres (3 et 5) comment est résolu le problème de présence de données atypiques en rendant l'agrégation des modèles de mélange plus robustes (ces approches forment les contributions de cette thèse).

2.9 Conclusion

Nous avons décrit dans ce chapitre, les fondements de la tâche de clustering dans le domaine de l'apprentissage non supervisé. Pour cela, nous avons montré les principaux types de méthodes de clustering. Nous nous sommes concentrés plutôt sur le clustering de modèles de mélanges probabilistes. En effet, les modèles de mélange ont été introduit dans un premier temps, ainsi que leur estimation. Puis, un exemple d'implémentation de l'algorithme *EM* dans le cadre de mélanges gaussiens, a été détaillé. Enfin, nous avons étudié le problème des données atypiques.

Ce chapitre fournit un état de l'art de la tâche de clustering, en particulier sur les méthodes basées sur les modèles de mélanges. La motivation derrière l'utilisation de ce type de méthodes a été exprimée dans la section (2.5), aussi bien dans le chapitre (1). Les modèles de mélange, ainsi que leur estimation, forment la partie principale de nos contributions dans cette thèse. Nous allons voir dans les chapitres suivantes, comment agréger les modèles de mélange, en particulier le mélange de *Student*, puis comment réaliser un clustering distribué en utilisant les mélanges gaussiens.

CHAPITRE 3

Agrégation robuste des modèles de mélange

3.1 Introduction

Ce chapitre présente une technique d'agrégation de modèles de mélange de la distribution de *Student*. Elle consiste à réduire un modèle de mélange de *Student* en un autre avec un nombre plus petit de composantes.

Nous avons montré dans le chapitre (1) la motivation derrière l'utilisation des modèles de mélange. Rappelons que ce type de modèle est considéré très utile pour réaliser l'apprentissage distribué. Récemment, l'agrégation des modèles de mélange a en particulier été très étudiée [114, 46, 42], à la fois en raison des coûts de transmission (seulement les paramètres de modèles sont communiqués), et peut être pour des problèmes de confidentialité des données.

La loi de *Student* a été présentée dans le domaine statistique comme une distribution robuste face aux données atypiques. Elle est spécifiée par un paramètre supplémentaire (degré de liberté) permettant de régler la robustesse du modèle. La distribution de *Student* est une généralisation de la loi gaussienne. Elle a été montrée dans la littérature comme une solution alternative au modèle gaussien, face à ce type de données [93].

Pour réaliser l'agrégation de mélanges de distribution de *Student*, nous avons adapté la technique de *Goldberger et al.* [47]. Cette méthode consiste à réduire des modèles de mélange gaussien. Elle se comporte comme la technique de *K-means* [63]. Elle se base seulement sur les paramètres de modèles sans accès ni aux données, ni à des vecteurs des attributs qui décrivent ces données. Elle est caractérisée donc par une complexité de calcul réduite (nombre limité des paramètres).

Plusieurs difficultés ont été rencontrées lors de l'application de la technique de *Goldberger et al.* [47] dans le cadre du mélange de *Student*. Cette méthode utilise la divergence de *KL* (*Kullback-Leibler*) pour mesurer la similarité entre le mélange original et réduit. Cependant, à notre connaissance, cette divergence n'existe pas analytiquement entre deux distributions de *Student* (section 3.3). Nous allons donc proposer une approximation de la distribution de *Student* par un mélange fini de composantes gaussiennes. Cette proposition nous permet d'utiliser l'une des approximations de la divergence du *KL* existante entre deux mélanges gaussiens.

Le reste de ce chapitre est organisé comme suit : d'abord, nous allons montrer comment approximer la distribution de *Student* par un mélange fini de gaussiennes. Nous décrivons ensuite la technique utilisée pour calculer la similarité entre deux mélanges de distribution de *Student* . Puis, nous allons détailler la technique de *Goldberger et al.* [47] pour la réduction d'un mélange gaussien, ainsi que notre adaptation de cette technique à la réduction des mélanges de distribution de *Student* . Enfin, nous allons valider notre proposition par des expériences.

3.2 Approximation de la distribution de *Student* par un mélange gaussien

Nous avons vu que la divergence *KL* entre deux modèles de *Student* est difficile à calculer, due à la complexité de la distribution de *Student*. Rappelons que la solution proposée consiste à approximer cette distribution par un mélange fini de gaussiennes.

Dans la suite, nous allons présenter dans un premier temps, la forme générale de la fonction de densité de distribution de *Student* , puis nous allons détailler notre technique pour approximer cette distribution par un mélange gaussien.

3.2.1 Distribution de *Student*

La distribution de *Student* est connue pour sa robustesse, due à sa large queue qui peut modéliser les données représentant des écarts importants par rapport à la moyenne. Elle est caractérisée par un paramètre nommé le degré de liberté jouant un rôle essentiel pour rejeter ce type de données.

La distribution de *Student* est une sommation infinie de la distribution gaussienne avec la distribution *Gamma*, elle s'écrit :

$$f(x|\mu,\Sigma,\nu) = \int \mathcal{N}(x|\mu,\Sigma/w)q(w)dw, \tag{3.1}$$

où $\mathcal{N}(x|\mu,\Sigma/w)$ est la distribution gaussienne définie par :

$$\mathcal{N}(x|\mu,\Sigma/w) = \frac{1}{(2\pi/w)^{d/2}|\Sigma|^{1/2}} e^{\frac{-w}{2}(x-\mu)^t\Sigma^{-1}(x-\mu)} \tag{3.2}$$

où μ est la moyenne et Σ/w est la matrice de covariance. $q(w)$ est la fonction de densité de la loi *Gamma* $G(\nu/2,\nu/2)$. Elle prend la forme suivante :

$$q(w,a,b) = \frac{a^b}{\Gamma(b)} e^{-aw} w^{b-1} I(w>0);(a,b>0), \tag{3.3}$$

où $I(w>0)=1$ si $w>0$ et zéro ailleurs. a et b sont respectivement le paramètre de forme et le paramètre d'intensité [15]. La figure (3.1) montre un exemple de la loi *Gamma*.

En remplaçant les équations (3.2) et (3.3) dans l'expression (3.1), alors la fonction de densité de la distribution de *Student* multivariée X, avec une moyenne μ, une matrice de covariance Σ et de degré de

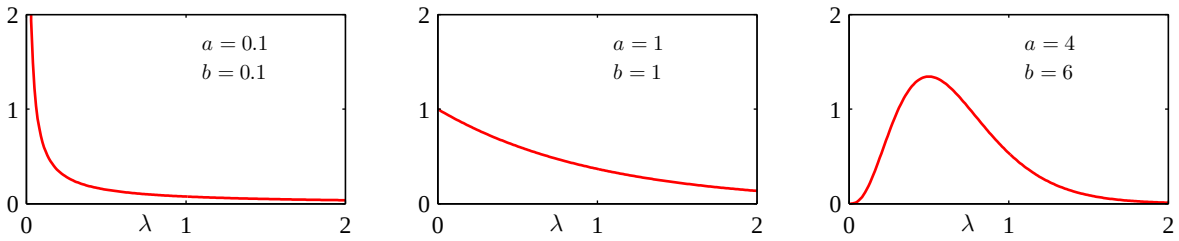


Figure 3.1 – La distribution *Gamma* $G(\lambda|a,b)$ avec différentes valeurs de paramètres a et b (extraite du [15]).

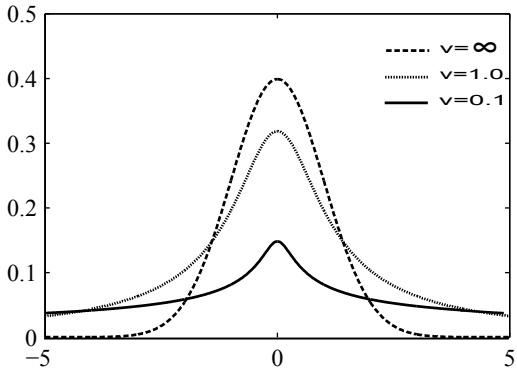


Figure 3.2 – Distribution de *Student* de $\mu = 0$ et $\sigma = 1$ avec différentes valeurs de degré du liberté ν. Si $\nu \to \infty$, la distribution de *Student* converge vers une loi gaussienne (figure extraite de [15]).

liberté $\nu \in (0, \infty]$ s'écrit :

$$f(x|\mu,\Sigma,\nu) = \frac{\Gamma(\frac{\nu+d}{2})|\Sigma|^{-1/2}}{(\pi\nu)^{\frac{d}{2}}\Gamma(\frac{\nu}{2})\{1+\delta(x,\mu,\Sigma)/\nu\}^{\frac{1}{2}(\nu+d)}}, \tag{3.4}$$

où la fonction

$$\delta(x,\mu,\Sigma) = (x-\mu)^t\Sigma^{-1}(x-\mu) \tag{3.5}$$

désigne la distance carré de *Mahalanobis* entre x et μ, et d est la dimensionnalité des données.

Si ν tend vers l'infini, X devient la loi normale multivariée avec le moyenne μ et la matrice de covariance Σ. La figure (3.2) montre la distribution de *Student* avec différentes valeurs du degré de liberté.

Les figures 3.3(a) et 3.3(b) montrent bien sur un échantillon de donnés contenant des données atypiques, la robustesse de la distribution de *Student* par rapport à la loi gaussienne.

La fonction de densité de la distribution de *Student* est considérée complexe (équation 3.4). Par conséquent, elle est parfois difficile à manipuler. Rappelons qu'il n'est pas facile de trouver une solution

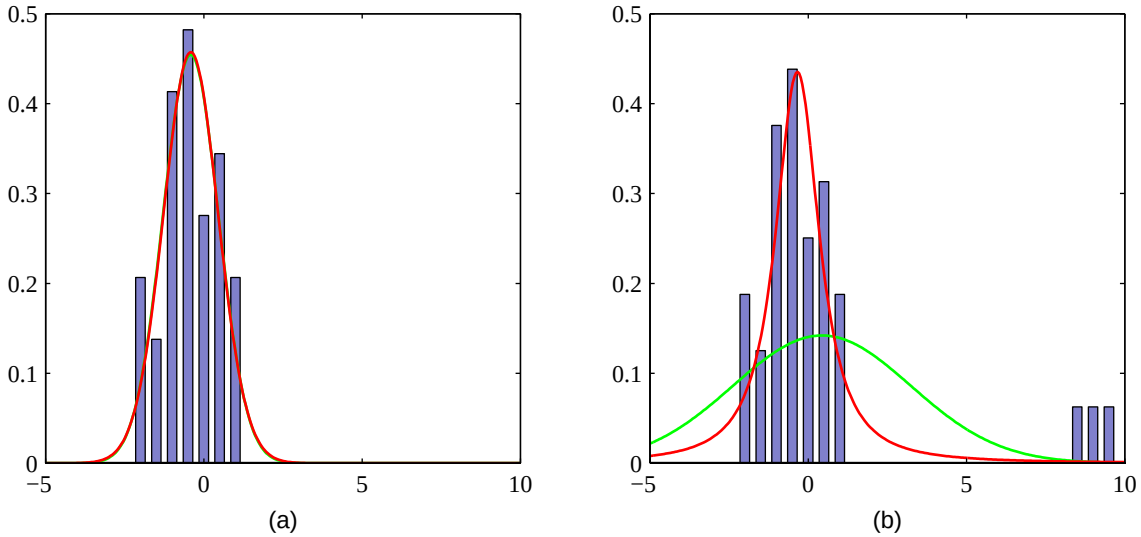


Figure 3.3 – Comparaison entre la distribution gaussienne et celle de *Student* sur un échantillon de 30 points générés à partir d'une distribution gaussienne. La figure (a) illustre l'histogramme de cet échantillon, ainsi que les deux courbes ajustées à ces points. Elles sont obtenues par la méthode de vraisemblance et elles apparaîssent presque collées. Ceci s'explique par le fait que la distribution gaussienne est considérée comme un cas particulier de la distribution de *Student*. Dans la figure (b), 3 points atypiques sont ajoutés à l'échantillon. Cette figure montre bien que la courbe gaussienne (grise) est bien affectée par les points atypiques, tandis que celle de *Student* (noire) reste identique (extraite de [15]).

directe pour calculer la divergence *KL* entre deux distributions de *Student* (voir section 3.3). C'est pour cette raison que nous proposons dans cette section une approximation de cette fonction par un mélange gaussien.

3.2.2 Distribution de *Student* comme une somme finie de gaussiennes

Nous avons vu dans la section précédente, que la fonction de densité de la distribution de *Student* est une sommation infinie de la distribution gaussienne. Ici, nous allons montré que cette fonction peut être approximée par une combinaison linéaire finie de densités gaussiennes de mêmes moyennes et de variances différentes. Les composantes de petites variances modélisent les points significatifs, tandis que les composantes de grandes variances représentent les données atypiques, si nécessaire. Le degré de liberté permet de définir le niveau de robustesse face aux données atypiques.

L'idée principale pour réaliser l'approximation de la distribution de *Student* a été inspirée de [15], en se basant sur le principe probabiliste suivant : étant donné une variable aléatoire X, l'espérance mathématique d'une fonction peut être approximée par la moyenne empirique des valeurs de X.

En probabilité, l'opération qui cherche à calculer la moyenne d'une fonction $f(x)$ sous une distribution de probabilité $p(x)$, est nommée l'espérance de $f(x)$. Elle est notée par $E(f)$ et définie comme suit [15] :

$$E(f) = \sum_x p(x)f(x) \tag{3.6}$$

si la distribution de variable x est discrète.
Dans le cas de variables continues, l'espérance prend la forme suivante :

$$E(f) = \int p(x)f(x)dx \tag{3.7}$$

L'espérance mathématique d'une fonction f peut être approximée par la moyenne empirique d'un échantillon, généré à partir de la probabilité de densité $p(x)$. En d'autres termes, si on considère l'existence de N points x_n générés à partir de la loi de probabilité $p(x)$, alors l'espérance de la fonction f peut être approximée par la sommation finie de $f(x)$ en ces points. Elle est calculée de la manière suivante [15] :

$$E(f) \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n) \tag{3.8}$$

Remarquons que la fonction de densité de distribution de *Student* peut s'exprimer comme une espérance mathématique de la fonction de densité de la loi gaussienne, sous la probabilité de densité de la loi *Gamma*. Ainsi, l'intégrale dans l'expression de la fonction de densité de la distribution de *Student* est calculée en fonction de la variable w qui suit la loi *Gamma* (équation (3.1)). Par conséquent, la fonction de densité de distribution de *Student* peut être approximée par la moyenne empirique de la fonction de densité de la loi gaussienne. Elle prend la forme suivante :

$$S(x) = \frac{1}{P}\sum_{p=1}^{P} \mathcal{N}(x, \mu, \frac{\Sigma}{w_p}) \tag{3.9}$$

où $w_1, \ldots, w_P$ sont générés à partir de la loi *Gamma* $G(w, \nu/2, \nu/2)$.

La distribution de *Student* est considérée maintenant comme une sommation finie d'un mélange de P gaussiennes. Les deux figures ((3.4) et (3.5)) illustrent un exemple d'approximation d'une distribution de *Student* par 10 gaussiennes. Le terme P dépend directement du degré de liberté de la distribution de *Student*. Si le degré de liberté ν est grand, le nombre de composantes gaussiennes nécessaires pour obtenir une bonne approximation est petit et vice et versa.

P est considéré comme un paramètre critique pour cette approximation. Sa valeur est déterminée expérimentalement (voir la section 3.6.1 pour plus de détails).

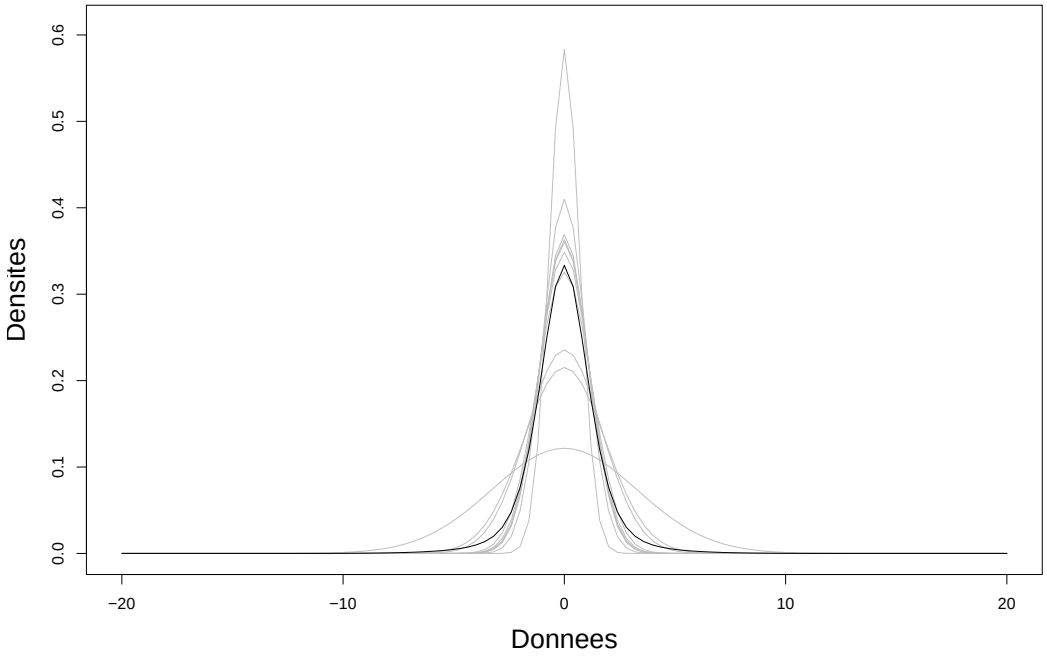


Figure 3.4 – Cette figure montre les courbes de 10 composantes gaussiennes (lignes en gris) et l'approximation obtenue (ligne en noire).

3.3 Similarité entre les modèles de mélange de la distribution de *Student*

Dans ce chapitre, nous avons besoin de mesurer la similarité entre le mélange original et le mélange réduit tout au long de l'exécution de notre méthode de réduction. Deux mesures sont très utilisées dans les techniques de réduction des modèles de mélange pour calculer la divergence entre deux distributions statistiques : la divergence de *Bregman* et la divergence de *KL*.

La divergence de *Bregman* est vue comme une généralisation de *KL*. Autrement dit : la divergence *KL* est une divergence de *Bregman*, utilisée pour mesurer la similarité entre deux distributions de la même famille exponentielle [87, 42]. Cette famille comprend plusieurs distributions (Gaussian, Laplacian, Poisson, Binomial, Bernoulli, etc). La divergence de *Bregman* peut être appliquée sur toutes les distributions de cette famille en exploitant leurs formes canoniques. Cependant à notre connaissance cette divergence n'existe pas entre les distributions de *Student*. Ceci peut être du à la densité de loi de *Student* qui est une sommation infinie de gaussiennes (c.à.d la densité de *Student* est complexe). Étant donné deux distributions de paramètres Θ_i et Θ_j respectivement, la divergence de *Bregman* est calculée comme suit :

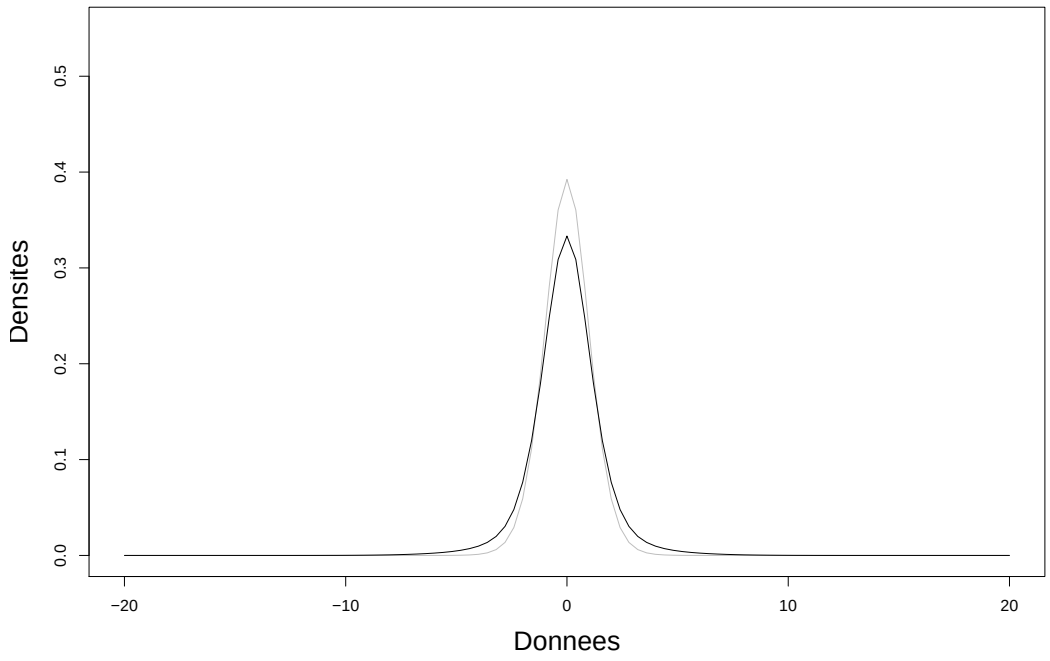


Figure 3.5 – Cette figure compare la courbe de la distribution de *Student* réelle (ligne en gris) et la courbe de notre approximation par gaussiennes (ligne en noire). Elle montre bien que les deux courbes sont proches : leurs queues sont lourdes dans les deux distributions.

$$D_F(\Theta_i||\Theta_j) = F(\Theta_i) - F(\Theta_j) - < \Theta_i - \Theta_j, \nabla F(\Theta_j) >, \tag{3.10}$$

où $<,>$ désigne le produit scalaire, ∇ est l'opérateur de dérivation et F est une fonction convexe (continue, dérivable) [87].

La divergence de *KL* entre deux distributions f_j et f_i est donc égale à la divergence de *Bregman* entre leurs paramètres permutés (Θ_i, Θ_j) : $D(f_j||f_i) = D_F(\Theta_i||\Theta_j)$. Nous utilisons donc la divergence de *KL* comme une mesure de similarité entre les distributions de *Student*. Mais le problème est que la divergence *KL* entre deux mélanges de distribution de *Student* n'existe pas aussi analytiquement. Cependant nous avons vu dans la section précédente, que la distribution de *Student* peut être approximée par un mélange de composantes gaussiennes. Nous proposons donc un *KL* analytique de *Student*, basé sur notre approximation de *Student* en gaussiennes.

La suite de cette section est ainsi organisée : tout d'abord, la divergence de *KL* est définie, puis les différentes méthodes développées pour approximer la divergence de *KL* entre deux mélanges gaussiens sont présentées. Enfin, la méthode choisie pour mesurer la divergence *KL* entre deux distributions de

Student est détaillée.

3.3.1 Divergence de *Kullback-Leibler*

La divergence de *KL*, connue aussi sous le nom de l'entropie relative, est très utilisée dans le domaine de statistique pour mesurer la similarité entre deux distributions de probabilité [59]. Elle est aussi très utilisée dans les domaines de reconnaissance de formes (images, sons) et l'apprentissage statistiques, pour, par exemple, mesurer la similarité entre deux modèles acoustiques [90], deux modèles d'images [45], ou pour réaliser un clustering des modèles [80, 102, 46].

Dans le cas des variables continues, elle est définie par :

$$D(f||g) = \int f(x) log \frac{f(x)}{g(x)} dx \tag{3.11}$$

où $f(x)$ et $g(x)$ sont deux fonctions de densités de probabilité.

Après avoir défini la divergence de *KL*, nous allons décrire dans la section suivante comment elle est appliquée pour mesurer la similarité entre deux mélanges gaussiens, et quelles sont les techniques proposées pour réaliser cet objectif.

3.3.1.1 Approximation de *Kullback-Leibler* entre deux mélanges gaussiens

Tout d'abord, supposons l'existence de deux mélanges gaussiens f et g qui prennent les formes suivantes :

$$\begin{aligned} f(x) &= \sum_{i=1}^{K} \pi_{f_i} \mathcal{N}(x, \mu_i, \Sigma_i) \\ g(x) &= \sum_{j=1}^{K'} \pi_{g_j} \mathcal{N}(x, \mu_j, \Sigma_j) \end{aligned} \tag{3.12}$$

La divergence entre deux composantes gaussiennes existe analytiquement en forme déterminée :

$$\begin{aligned} KL(f_i||g_j) = \quad & \tfrac{1}{2}[log \tfrac{|\Sigma_{g_j}|}{|\Sigma_{f_i}|} + Tr|\Sigma_{g_j}^{-1}\Sigma_{f_i}| \\ & -d + (\mu_{f_i} - \mu_{g_j})^t \Sigma_{g_j}^{-1} (\mu_{f_i} - \mu_{g_j})] \end{aligned} \tag{3.13}$$

où d est la dimension de l'espace des données et Tr désigne la trace de la matrice. Les ensembles $\{\mu_{f_i}, \Sigma_{f_i}\}$ et $\{\mu_{g_j}, \Sigma_{g_j}\}$ désignent respectivement les moyennes et les covariances des composantes gaussiennes f_i et g_j .

Pour calculer la divergence entre deux mélanges gaussiens, il n'existe pas de solution en forme déterminée. Pour cela, plusieurs techniques sont proposées pour calculer des approximations de *KL* [59]. Elles

peuvent être divisées en deux catégories : des méthodes basées sur l'échantillonnage de points et d'autres sur les paramètres de modèles.

Méthodes basées sur l'échantillonnage : ce type de méthodes calcule la divergence de *KL* en se basant sur des échantillons des points. Ces points sont générés directement à partir des mélanges f (méthode de *Monte Carlo*) ou calculés à partir des matrices de covariances des composantes gaussiennes de f (méthode de *unscented transformation*). Les points générés sont ensuite utilisés pour calculer la divergence de *KL* à partir de l'équation (3.11). Les méthodes basées sur les points produisent des résultats nettement meilleurs que ceux obtenus par les méthodes basées sur les paramètres [59], en particulier si le nombre de points est élevé. Cependant, elles ont des complexités de calculs beaucoup plus élevées, puisque l'équation (3.11) implique d'itérer sur toutes les données. Deux méthodes de ce type sont proposées : la méthode de *Monte Carlo* et et la méthode de *unscented transformation* [45, 59].

Méthode de Monte Carlo : c'est la méthode la plus connue [59]. Elle réalise les meilleurs résultats parmi toutes les méthodes. Elle consiste à générer un échantillon de n points de modèle f et elle calcule la divergence de *KL* comme suit :

$$D_{MC} = \frac{1}{n}\sum_{p=1}^{n} log(f(x_p)/g(x_p)) \tag{3.14}$$

Méthode de unscented transformation : cette méthode à la base consiste à calculer les statistiques (espérance, covariance) d'une variable aléatoire qui suit une transformation non linéaire [70]. La divergence *KL* peut s'écrire comme $D(f||g) = \sum_i \pi_{f_i} E_f[h]$ où $h = log(f/g)$ est une fonction non linéaire. Le but ici est d'estimer $E_f[h]$ l'espérance de h en fonction de points *sigma*. Ces points (un échantillon de taille faible) sont obtenus à partir des valeurs propres et des vecteurs propres de matrices de covariances des composantes gaussiennes de f [45, 59]. Cette méthode donne des résultats légèrement moins bons qu'avec la méthode de *Monte Carlo*. Elle nécessite un temps d'exécution élevé dû au calcul des valeurs et vecteurs propres.

Méthodes basées sur les paramètres des modèles : ces méthodes calculent des approximations de *KL* directement à partir des paramètres des modèles, sans accès aux données. En comparaison avec les méthodes basées sur l'échantillonnage de points, elles donnent le meilleur compromis entre la qualité de résultats et le coût de calculs. En effet, le nombre de paramètres est beaucoup plus faible que celui des données, ce qui entraîne une complexité de calcul beaucoup plus réduite. Dans ce chapitre, nous nous intéressons plutôt à ce genre de méthodes, dans lesquelles plusieurs techniques peuvent être distinguées [59] :

Méthodes basées sur des composantes représentatives : elles consistent à remplacer chaque mélange gaussien par une seule composante gaussienne représentative, et de calculer la divergence entre deux mélanges gaussiens comme étant le *KL* entre les gaussiennes obtenues. Les deux composantes représentatives peuvent être les moyennes des deux mélanges ou la paire la plus proche dans les deux mélanges. Ces méthodes sont simples à calculer, mais elles produisent de mauvais résultats en comparaison avec les autres méthodes existantes. Cela s'explique par le fait qu'un mélange est remplacé seulement par une seule composante, ce qui entraîne une perte d'information importante.

Méthodes basées sur le produit de mélanges : la divergence *KL* peut être exprimée comme la différence entre la fonction de vraisemblance de f et celle de g. Elle s'écrit comme suit : $D(f||g) = L_f(f) - L_f(g)$. Cette méthode cherche à trouver des bornes supérieures pour ces fonctions, et elle calcule l'approximation de *KL* en fonction du produit de composantes gaussiennes de f et g. Étant donné que cette méthode calcule les bornes supérieures des fonctions de vraisemblance de f et g plutôt que des approximations exactes, elle produit aussi des mauvais résultats.

Méthode de match bound : la méthode de *match bound* [45] définie d'abord une fonction de correspondance (affectation binaire) entre les composantes de f et g (voir la figure (3.6)). Puis, elle utilise les résultats fournis par cette fonction pour calculer l'approximation de *KL*. La méthode de *match bound* est choisie dans ce chapitre pour calculer la divergence entre deux mélanges gaussiens. L'utilisation de cette méthode est motivée par la qualité de ses résultats mais aussi par sa complexité de calcul réduite (utilisation d'un nombre très limité de paramètres).

Méthode variationnelle : cette méthode calcule aussi l'approximation de *KL* comme la différence entre la fonction de vraisemblance de f et celle de g, mais en utilisant les paramètres variationnelles. Ces paramètres sont introduits dans les fonctions de vraisemblance, et ils sont définis comme suit : $(\phi_b, \psi_a) > 0$, $\sum_b \phi_b = 1$, $\sum_a \psi_a = 1$. Ils sont obtenus par la maximisation de fonctions de vraisemblance. D'après les travaux de [59], cette méthode donne de meilleurs résultats que celles basées sur les paramètres.

Pour résumer, les méthodes *match bound* et *variationnelle* sont les meilleures parmi les méthodes paramétriques. La méthode de *match bound* fournit des résultats proches de ceux de la méthode variationnelle (surtout si le nombre de composantes de f et g est différent), et elle est meilleure que les autres méthodes basées sur les paramètres. Elle est préférée ici à la méthode variationnelle, à cause de sa fonction de correspondance qui nous sera utile dans la méthode de réduction de mélange de distribution de *Student* proposée, en particulier dans la phase mise à jour des paramètres (voir la section (3.5)).

3.3.2 Méthode de *match bound* et application sur deux composantes de *Student*

Dans cette section, nous décrivons analytiquement la méthode *match bound*, choisie pour calculer la divergence *KL* entre deux composantes de *Student* .

Méthode de *match bound*

Supposons l'existence de deux composantes de *Student* f et g approximées respectivement par des mélanges de K et $K^{'}$ gaussiennes. L'idée principale de cette méthode est de définir une fonction de correspondance $m : \{1, \cdots, K\} \rightarrow \{1, \cdots, K^{'}\}$, entre les K composantes de f et les $K^{'}$ composantes de g comme suit :

$$m(i) = \underset{j}{\operatorname{argmin}}\, D(f_i||g_j) - log(\pi_{g_j}). \tag{3.15}$$

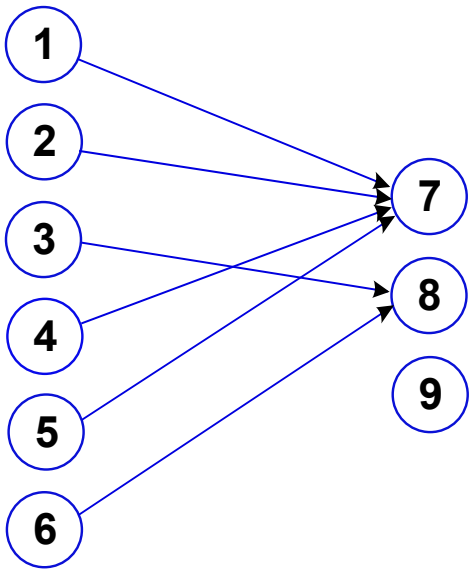


Figure 3.6 – Cette figure présente la correspondance entre deux mélanges composés respectivement de 6 et 4 composantes gaussiennes. Notons que cette fonction de correspondance est non surjective, comme dans ce cas là, où la composante 9 n'est pas liée à aucune composante du premier mélange.

La fonction m correspond aux composantes les plus proches dans les mélanges f et g. f_i et g_i sont respectivement les composantes gaussiennes de f et g. $D(f_i||g_j)$ est la divergence *KL* définie par l'équation (3.11).

L'approximation de la divergence de *KL* par la méthode de *match bound* est obtenue par la formule suivante :

$$KL_{match}(f||g) = \sum_i \pi_{f_i} \left(D(f_i||g_{m(i)}) + log \frac{\pi_a}{\pi_{g_{m(i)}}} \right), \tag{3.16}$$

où π_{f_i} et $\pi_{g_{m(i)}}$ sont respectivement les probabilités à priori des composantes f_i et $g_{m(i)}$.

Exemple de *KL* entre deux composantes de *Student*

L'approximation de la divergence de *KL* entre deux composantes de *Student* revient donc à calculer la divergence de *KL* entre deux mélanges de gaussiennes. Pour mesurer la similarité entre deux distributions de *Student* (ou bien entre deux mélanges de *Student*), il faut donc appliquer les étapes suivantes :

– approximer chaque composante par un mélange gaussien. Chacun mélange est composé par un nombre P de composantes de même moyenne. Par exemple, les composantes f_i et g_i dans la figure (3.7) sont approximées chacune par 3 composantes gaussiennes. Notons que chaque com-

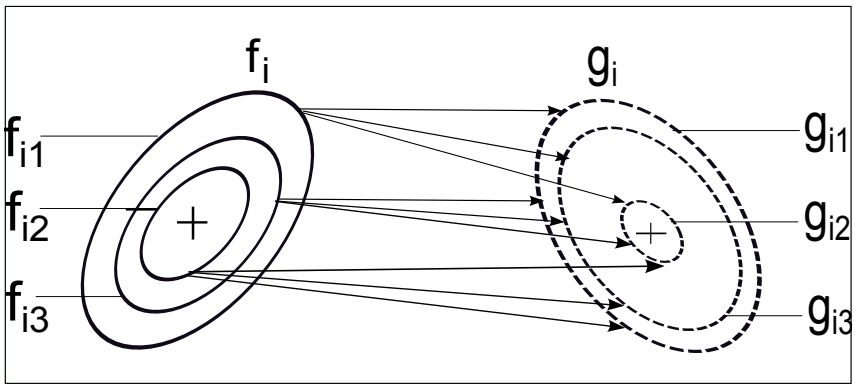


Figure 3.7 – Exemple de la divergence de *KL* entre deux composantes de distribution de *Student* f_i et g_i. Chaque composante de *Student* a été approximée par un mélange de gaussiennes de $P = 3$ composantes. Pour calculer la divergence de *match bound*, toutes les combinaisons de *KL* entre les composantes gaussiennes de f_i et g_i sont calculées. La meilleure m minimisant le *KL* entre les gaussiennes est utilisée après pour calculer l'équation (3.16).

posante de *Student* peut avoir un nombre différent de gaussiennes ;

– calculer la divergence *KL* de *match bound* entre les mélanges obtenus. Le but est de trouver parmi toutes les combinaisons de *KL* entre les composantes gaussiennes celle (m) qui minimise l'équation (3.16). Par exemple, dans la figure (3.7) les *KL* entre les deux ensembles $\{f_{i1}, f_{i2}, f_{i3}\}$ et $\{g_{i1}, g_{i2}, g_{i3}\}$ sont calculés afin de trouver la meilleure correspondance m définie par l'équation (3.15) ;

Ces deux opérations sont répétées entre chaque paire de composantes de *Student* de mélange original et réduit.

Après avoir défini la métrique entre deux composantes de *Student*, nous pouvons appliquer maintenant notre méthode pour réduire un mélange de *Student* .

3.4 Algorithme général de la réduction des modèles de mélange

Dans cette section, nous présentons d'abord un état de l'art sur l'agrégation de modèles de mélange. Puis, nous détaillons dans un deuxième temps la technique de *Goldberger et al.* [47] utilisée comme une méthode de base dans cette thèse pour réduire les mélanges.

3.4.1 Agrégation de modèles de mélange

Le problème de la réduction d'un modèle de mélange consiste à transformer un modèle de mélange f en un autre mélange g avec un nombre plus petit de composantes. Le mélange réduit g obtenu est considéré comme la meilleure approximation de f selon un critère de similarité donné. Ce critère sert à

minimiser la divergence entre les deux mélanges original et réduit.

La façon la plus simple pour réduire un modèle de mélange consiste à ré-estimer le modèle de mélange directement à partir des données originales, en précisant à priori le nombre de composantes que l'on souhaite obtenir. Cependant, cette technique ne peut pas être appliquée pour plusieurs raisons : la complexité de calcul des modèles de mélange est considérée élevée, en particulier si la quantité des données étudiées est large. Ainsi, les données originales ne peuvent pas être disponibles.

Dans le cas des modèles distribués, l'agrégation des modèles de mélange (fournis par les sources distribuées) peut être obtenue simplement par la somme pondérée d'un ensemble de mélanges. Mais ceci peut conduire à un grand nombre des composantes inutiles dans le modèle [19]. L'objectif est de réaliser plutôt une estimation d'un mélange parcimonieux à partir d'un ensemble de modèles. La propriété de parcimonie, est spécialement importante si par exemple les agrégations de mélanges se succèdent les uns après les autres.

Les méthodes d'agrégation peuvent être divisées en deux catégories : les méthodes basées sur l'échantillonnage des données [114, 80, 46], et les méthodes qui utilisent seulement les paramètres des modèles [86, 42, 43, 47]. Par la suite, nous allons décrire quelques unes de ces méthodes.

Méthodes basées sur l'échantillonnage de données

Les auteurs dans [80] cherchent à trouver le modèle réduit en deux étapes. Dans la première étape, les modèles locaux obtenus sur les sites répartis sont transmis vers un site central afin de calculer le modèle moyen (somme pondérée des modèles). Dans la deuxième étape, un nouveau modèle est construit à partir d'un échantillon des données, généré à partir d'un modèle moyenne, en utilisant une technique usuelle pour l'estimation des modèles de mélange comme *EM*. Le modèle réduit est obtenu comme celui qui a le minimum de divergence de *KL* avec le modèle moyen.

Dans [114], une méthode hiérarchique ascendante a été utilisée pour agréger les modèles de mélange. Elle consiste à estimer un nouveau modèle à chaque niveau hiérarchique l à partir d'un échantillon généré à partir de modèle du niveau $l-1$.

Les travaux de [46] se basent sur la méthode de *unscented transformation* [45] pour réaliser la réduction des modèles de mélange. L'estimation de modèle réduit est effectuée à partir d'un ensemble de points (nommés points de *sigma*) calculés selon cette technique, en minimisant le critère de *unscented transformation* entre deux mélanges gaussiens. Un inconvénient majeur de cette méthode est qu'elle nécessite de calculer les vecteurs propres des matrices des covariances. Cette opération consomme un coût élevé en terme de temps de calcul [42].

Les méthodes qui reposent sur l'échantillonnage des points peuvent être coûteuses, surtout si la dimension des données étudiées est élevée.

Méthodes basées sur les paramètres de modèles

Ces méthodes se basent seulement sur les paramètres de modèles pour réaliser l'agrégation, sans accédée aux données réelles. Elles sont caractérisées par plusieurs avantages :

- elles sont spécifiées par une complexité de calcul réduite (utilisation d'un nombre limité de paramètres) ;

- elles sont donc très utiles pour les applications distribuées (coût de transmission faible) et pour le problème de confidentialité des données ;

Rappelons qu'il existe deux critères de divergence qui sont très utilisés dans ces méthodes : la divergence de *KL* [59] et la divergence de *Bregman* [43, 87]. Notons qu'une fois la divergence entre les composantes définie, n'importe quel type de clustering (*hiérarchique*, *K-means*, *EM*) peut être généralisé pour être appliqué sur des composantes plutôt que les données. Par la suite, plusieurs exemples pour la réduction des modèles de mélange sont décrits.

Des méthodes ont exploité la forme canonique de la famille exponentielle (Gaussian, Laplacian, Poisson, binomial, Bernoulli, etc) afin d'utiliser la divergence de *Bregman* [86, 42, 43]. En se basant sur cette divergence, les méthodes proposées peuvent être appliquées pour réduire n'importe quel mélange appartenant à cette famille.

Par exemple, les travaux des [86, 42] ont présenté des techniques de simplification basées sur la méthode de *K-means*. Ces techniques (comme *K-means*) commencent donc par une étape d'initialisation. Puis, elles itèrent entre la phase d'affectation et la mise à jour des paramètres. Les objets manipulés sont les composantes de mélanges. Une composante est affectée à un cluster donné si elle a le minimum de divergence de *Bregman* avec le centre du cluster.

De la même façon, les auteurs de [42] ont utilisé la forme canonique des distributions exponentielles et la divergence de *Bregman* pour proposer deux techniques d'agrégation. La première méthode est une adaptation de l'algorithme EM appliquée aussi directement sur les paramètres des composantes. La deuxième est une technique d'agrégation hiérarchique. Elle consiste à fusionner à chaque fois les deux composantes les plus proches entre elles.

Les travaux cités ci-dessus peuvent donc être réalisés sur toutes les distributions de la famille exponentielle en utilisant la divergence de *Bregman*. Cependant en rappelant la section (3.3), à notre connaissance, cette divergence n'existe pas entre les distributions de *Student* (densité complexe : sommation infinie de gaussiennes). Par conséquent, aucune méthode parmi elles ne peut être appliquée directement sur un modèle de *Student*.

D'autres méthodes utilisent plutôt la divergence de *KL* comme une mesure de similarité entre les composantes de modèle. Par exemple, les auteurs dans [102] cherchent à combiner chaque fois les deux composantes les plus similaires. Ils calculent le *KL* entre le mélange, avant et après le fusionnement de chaque paire de composantes. Cette méthode réalise seulement la réduction d'un mélange gaussien. Elle n'a pas été appliquée à un mélange de *Student*.

Des techniques variationnelles pour réduire les mélanges de gaussiennes et les mélanges d'ACP (Principal Components Analysis en anglais) ont été développées dans notre équipe [19] [18]. Ces tra-

vaux sont effectués en parallèle avec notre méthode proposée. Elles consistent à adapter l'algorithme *EM* variationnel à des entrées constituées de composantes (gaussiennes ou ACP) au lieu de données. L'avantage principal de ces approches est que la complexité des modèles (nombre correct des composantes) à estimer est calculée implicitement. Cependant, elles sont implémentées pour être appliquées sur des mélanges gaussiennes et ACP et non pas sur un mélange de *Student*.

Les travaux réalisés dans *Goldberger et al.* [47], présentent aussi une technique pour répondre au problème de réduction de mélanges gaussiens. Pour chercher le mélange réduit optimal g, les auteurs ont proposé un algorithme itératif qui se déroule de la même façon que l'algorithme *K-means*, mais sur les composantes gaussiennes au lieu des données. Similaire aux méthodes utilisant la divergence *KL*, la technique de *Goldberger et al.* a été appliquée pour réduire seulement un mélange gaussien.

Pour résumer, à notre connaissance, aucune méthode parmi celles basées sur les paramètres de modèles ne traite le problème de réduction d'un mélange de *Student*. Pour cette raison, nous avons proposé une technique pour calculer le *KL* entre deux distributions de *Student*. En se basant sur cette divergence, nous pouvons réaliser la réduction de plusieurs manières, en utilisant par exemple, soit un algorithme hiérarchique, soit un algorithme de partitionnement (*K-means*, *EM)*. Nous avons choisi la technique de *Goldberger et al.* pour l'adapter en termes de mélange de *Student*. Rappelons que cette méthode fonctionne de la même façon que l'algorithme de *K-means*. Elle est donc facile à manipuler et est caractérisée par une complexité de calcul réduit (de l'ordre O(KN) dans le cas de données, avec K le nombre de clusters et N le nombre de données).

3.4.2 Algorithme de réduction d'un mélange gaussien

Dans cette section, nous présentons la technique de *Goldberger et al.* [47] pour la réduction d'un mélange gaussien. Elle consiste à réduire un modèle de mélange f en un autre modèle g en regroupant les composantes similaires.

Rappelons que cette méthode fonctionne comme l'algorithme *K-means* [63]. Elle itère entre deux étapes. Dans la première étape, la divergence *KL* entre les deux mélanges est calculée. Le but est d'associer à chaque composante de g, les composantes de f les plus proches. Autrement dit, elle sert à trouver la meilleure association entre les composantes de f et g minimisant le *KL* entre elles. Dans la deuxième étape, chaque composante de g est mise à jour à partir de composantes correspondantes de f, en se basant sur les résultats obtenus dans la première étape.

Similaire à toutes les méthodes d'agrégation basées sur les paramètres, la méthode de *Goldberger et al.* est caractérisée par les mêmes avantages (complexité de calcul faible, pertinente pour les applications distribuées, ..). Pour la suite, nous allons décrire cette méthode pour résumer un modèle de mélange gaussien.

Supposons l'existence d'un mélange f composé de K d-dimensionnel composantes gaussiennes :

$$f(x) = \sum_{i=1}^{K} \pi_{f_i} f_i(x). \tag{3.17}$$

Le mécanisme central de leur technique de regroupement de composantes, consiste à approximer la divergence de *KL* entre deux mélanges f et g comme suit :

$$d(f,g) = \sum_{i=1}^{K} \pi_{f_i} \min_{j=1}^{K'} KL(f_i||g_j) \tag{3.18}$$

où K et $K^{'}$ désignent respectivement le nombre de composantes de f et g, avec $K^{'} < K$. π_{f_i} est la probabilité à priori de la composante f_i. Elle itère entre les deux étapes suivantes.

1. **Divergence entre les composantes de modèle original et réduit**

   Selon *Goldberger et al.* [47], le but de cette étape est de chercher parmi toutes les combinaisons possibles de correspondances (affectation binaire) entre les composantes de f et de g, celle qui minimise l'équation (3.18). Notons par S l'ensemble de toutes les correspondances possibles ϕ de $\{1,\cdots,K\} \rightarrow \{1,\cdots,K^{'}\}$ et par m la correspondance optimale, alors le critère suivant doit être optimisé :

   $$\begin{aligned} d(f,g) = &\quad d(f,g,m) \\ &= \sum_{i=1}^{K} \pi_{f_i} KL(f_i||g_{m(i)}) \end{aligned} \tag{3.19}$$

   où $m = \arg\min_{\phi} \quad d(f,g,\phi)$.

   À la fin de cette étape, chaque composante f_i de f doit être liée à une seule composante de g. La composante de g correspondante $g_{m(i)}$ est obtenue selon m. Notons qu'à l'inverse des composantes de f, il est possible qu'une composante de g puisse ne puisse être liée à aucune composante de f. Une telle composante représente une grande divergence par rapport aux composantes de f (la composante 9 dans la figure 3.6).

2. **Mise à jour des paramètres de modèle réduit**

   Dans la deuxième étape, les paramètres du modèle g sont mis à jour à partir des paramètres du modèle original f en se basant sur la correspondance m, calculée dans la première étape. Les paramètres de g sont mis à jour comme suit :

   $$\mu_{g_j} = \quad \frac{1}{\pi_{g_j}} \sum_{i \in m^{-1}(j)} \pi_{f_i} \mu_{f_i} \tag{3.20}$$

   $$\begin{aligned} \Sigma_{g_j} = &\quad \frac{1}{\pi_{g_j}} \sum_{i \in m^{-1}(j)} \pi_{f_i} \\ &\left((\Sigma_{f_i} + (\mu_{f_i} - \mu_{g_j})(\mu_{f_i} - \mu_{g_j})^T)\right) \end{aligned} \tag{3.21}$$

   où $\pi_{g_j} = \sum_{i \in m^{-1}(j)} \pi_{f_i}$.

m^{-1} est la fonction inverse de m définie en première étape. Par exemple, $m^{-1}(j)$ contient toutes les composantes de f qui sont attachées à la composante g_j de g, selon m. La mise à jour (moyenne, covariance) de la composante g_j est donc réalisée à partir de celles de f présentées dans $m^{-1}(j)$. Cependant, si $m^{-1}(j)$ est vide, ça veut dire que la composante g_j n'est associée à aucune des composantes de f. Dans ce cas, cette composante n'est pas modifiée (même moyenne et covariance).

Ces deux étapes sont itérées jusqu'à la convergence du critère (3.18).

Notons que comme notre technique d'agrégation se déroule de la même façon que l'algorithme *K-means*. Elle présente donc les mêmes inconvénients :

– le nombre de composantes $K^{'}$ du modèle réduit doit être précisé à l'avance. Cela représente le seul paramètre critique de notre algorithme ;

– les résultats de l'agrégation dépendent directement de la phase d'initialisation des paramètres. Nous allons voir dans la section suivante comment est résolu ce problème par la technique de *k-means++* [7] ;

La figure (3.8) illustre la technique de *Goldberger et al.* [47] pour la réduction d'un modèle de mélange gaussien. L'algorithme (5) résume ainsi les différentes étapes de *Goldberger et al.* [47].

Dans la section suivante, nous allons expliquer, comment la technique de *Goldberger et al.* [47] peut être adaptée à la réduction de mélange de distribution de *Student* .

3.5 Algorithme de réduction de mélanges de distribution de *Student*

Nous avons décrit ci-dessus, la technique de *Goldberger et al.* [47] pour réduire un mélange gaussien. Nous proposons dans cette section, une adaptation des ses étapes en termes de mélange de distribution de *Student*. Pour cette raison, nous avons changé le critère à optimiser, en remplaçant la divergence de *KL* entre deux gaussiennes, par celle de *match bound* entre deux composantes de *Student* (approximées par des gaussiennes). Dans la suite, nous allons détailler notre proposition pour réduire un mélange de distribution de *Student*.

Supposons l'existence d'un mélange f, composé de K composantes de distribution de *Student* :

$$f(x) = \sum_{i=1}^{K} \pi_{f_i} f_i(x). \tag{3.22}$$

Nous avons vu, que chaque composante de distribution de *Student* peut être approximée par un mélange gaussien. Le mélange f devient donc un mélange de mélanges gaussiens prenant la forme suivante :

$$f(x) = \sum_{i=1}^{K} \pi_{f_i} f_i(x) = \sum_{i=1}^{K} \pi_{f_i} \left(\sum_{p=1}^{P} \pi_{f_{ip}} f_{ip} \right), \tag{3.23}$$

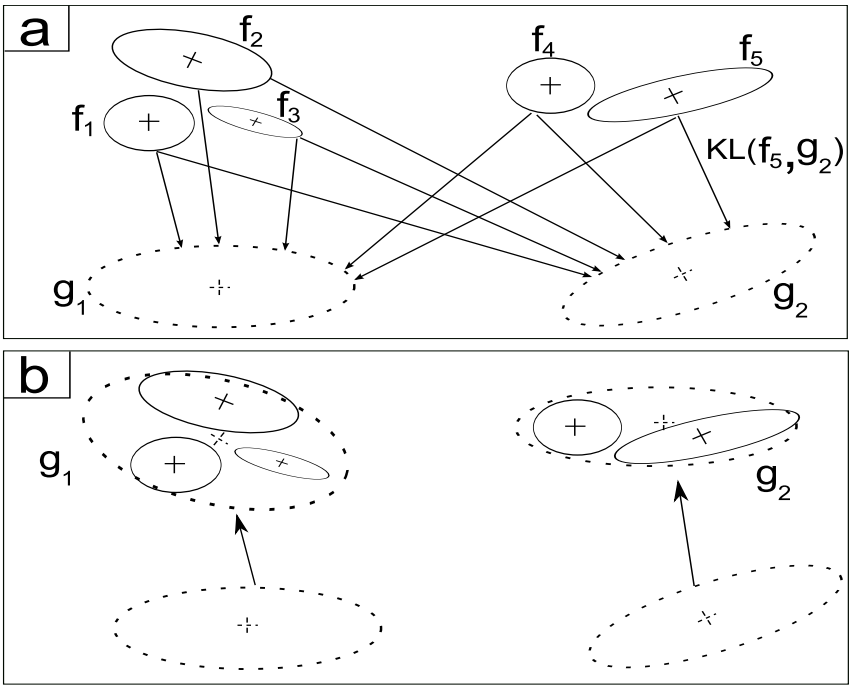


Figure 3.8 – Réduction de modèles de mélange en utilisant la technique de *Goldberger et al.* [47]. Les ellipses continues et pointillées représentent respectivement les modèles de f et g. La figure (a) montre la première étape de l'algorithme où les divergences entre les composantes de f et g sont calculées. La meilleure correspondance obtenue m, consiste à associer, par exemple, l'ensemble $\{f_1, f_2, f_3\}$ à g_1 et les composantes restantes à g_2. La figure (b) présente les mises à jour des paramètres de modèle réduit g en se basant sur la fonction de correspondance m, et en minimisant le critère (3.18). Les flèches ici désignent le déplacement des composantes de g après les mises à jour.

où P est le nombre de composantes gaussiennes, utilisées pour approximer les composantes de distribution de *Student*. La valeur de P est déterminée par la suite via notre expérience (voir la section 3.6.1).

Le mélange réduit g composé par un nombre $K^{'} < K$ de composantes est transformé aussi en un mélange de mélanges gaussiens :

$$g(x) = \sum_{j=1}^{K^{'}} \pi_{g_j} g_j(x) = \sum_{j=1}^{K^{'}} \pi_{g_j} \left(\sum_{p=1}^{P} \pi_{g_{jp}} g_{jp} \right), \tag{3.24}$$

Le nombre de composantes gaussiennes P, pour toutes les composantes g_j, est initialisé par la plus grande valeur utilisée pour approximer les composantes f_i de f. Ce choix est expliqué par le fait que la composante de distribution de *Student*, avec le degré de liberté le plus petit, est choisi. Dans le cas contraire, la composante est représentée par un nombre plus petit de composantes. Ceci peut entraîner une perte d'informations (représentation mauvaise).

Entrées : Deux mélanges gaussiens f et g , composés respectivement de K et $K^{'}$ composantes $(K > K^{'})$.

répéter

- Trouver la meilleure correspondance m, puis calculer l'approximation de la divergence de *KL* entre les composantes de f et g comme suit :

$$\begin{aligned} d(f,g) = \quad & d(f,g,m) \\ = & \sum_{i=1}^{K} \pi_{f_i} KL(f_i||g_{m(i)}) \end{aligned}$$

- Mettre à jour les paramètres des composantes de g seulement à partir des paramètres de f en se basant sur la fonction de correspondance m :

$$\begin{aligned} \mu_{g_j} = & \quad \tfrac{1}{\pi_{g_j}} \textstyle\sum_{i \in m^{-1}(j)} \pi_{f_i} \mu_{f_i} \\ \Sigma_{g_j} = & \quad \tfrac{1}{\pi_{g_j}} \textstyle\sum_{i \in m^{-1}(j)} \pi_{f_i} \\ & \left((\Sigma_{f_i} + (\mu_{f_i} - \mu_{g_j})(\mu_{f_i} - \mu_{g_j})^T)\right) \end{aligned}$$

jusqu'à *le critère de convergence (3.18) n'est pas encore minimisé*;

Sorties : Mélange g, la réduction optimale de f

Algorithme 5: Algorithme de réduction d'un mélange gaussien

De la même façon que la réduction d'un mélange gaussien, l'algorithme proposé pour réduire le mélange d'une distribution de *Student* se déroule d'une façon itérative en alternant entre les deux étapes principales suivantes.

1. **Divergence entre les composantes de modèle original et réduit**

   Rappelons que cette étape consiste à chercher parmi toutes les combinaisons possibles des correspondances (affectation binaire) entre les composantes de *Student* de f et de g, celle minimisant la divergence entre ces deux mélanges. La seule différence dans cette étape par rapport à la réduction de mélange gaussien est que la divergence *KL* est remplacée par celle de $KL_{matchbound}$ (équation 3.16). Le critère suivant doit ainsi être optimisé :

$$\begin{aligned} d(f,g) = \quad & d(f,g,m) \\ = & \sum_{i=1}^{K} \pi_{f_i} KL_{match}(f_i||g_{m(i)}) \end{aligned} \tag{3.25}$$

   où $m = \arg\min_{\phi} \quad d(f,g,\phi)$, et $\phi : \{1,\cdots,K\} \rightarrow \{1,\cdots,K^{'}\}$ est la fonction de correspondance entre les composantes de f et g. f_i et $g_{m(i)}$ représentent respectivement des composantes de

Student (approximées par des mélanges gaussiens) de f et g, liées par la fonction de correspondance m.

2. **Mise à jour des paramètres de modèle réduit**

Cette étape consiste à mettre à jour le modèle g à partir des paramètres du modèle f uniquement en se basant d'abord sur la fonction de correspondance m entre les composantes de distribution de *Student*, puis sur la fonction de correspondance entre les composantes gaussiennes, réalisées par l'équation (3.15). Cette dernière fonction est toujours nommée $m^{'}$, elle est obtenue lors du calcul de la divergence entre deux composantes de distribution de *Student*, en utilisant la méthode de *match bound*. m et $m^{'}$ sont ainsi respectivement les fonctions de correspondance entre les composantes de *Student* de f et g, et les composantes gaussiennes de f_i et g_j. Un exemple d'application simplifié est illustré dans la figure (3.9). Les composantes de distribution de *Student*, f_1 et f_2 du modèle original f, sont liées aux composantes g_1 du modèle réduit g, selon la fonction de correspondance m. Ensuite, les composantes gaussiennes de f_1 et f_2 sont attachées aux composantes gaussiennes de g_1 selon $m^{'}$. Les paramètres de composantes gaussiennes du modèle g sont mis à jour comme suit :

$$\mu_{g_j} = \frac{1}{\pi_{g_j}} \sum_{i \in m^{-1}(j)} \pi_{f_i} \mu_{f_i} \tag{3.26}$$

$$\Sigma_{g_{jp}} = \frac{1}{\pi_{g_j}} \sum_{i \in m^{-1}(j)} \pi_{f_i} \left(\frac{1}{|m'^{-1}(p)|} \sum_{l \in m'^{-1}(p)} (\Sigma_{f_{il}} + (\mu_{f_i} - \mu_{g_j})(\mu_{f_i} - \mu_{g_j})^T) \right) \tag{3.27}$$

où $\pi_{g_j} = \sum_{i \in m^{-1}(j)} \pi_{f_i}$ et $\Sigma_{g_{jp}}$ est la $p$ème covariance de composante g_j.

La moyenne de chaque composante μ_{g_j} est calculée directement à partir de celles du modèle f en se basant seulement sur la fonction m. Rappelons que selon notre approximation de la distribution de *Student* par un mélange gaussien, la moyenne de cette composante est identique pour chaque gaussienne. Il suffit donc de mettre à jour la moyenne de chaque composante de *Student*, et d'affecter ensuite cette valeur à toutes les moyennes de ses composantes gaussiennes associées. Par exemple, dans la figure (3.9), la moyenne μ_{g_1} de la composante g_1 est calculée à partir des moyennes de f_1 et f_2 selon la fonction m. Rappelons que par construction toutes les composantes gaussiennes g_{1p} possèdent la même moyenne μ_{g_1}.

Pour les matrices de covariances, chaque matrice de covariance $\Sigma_{g_{jp}}$ est mise à jour en fonction de m et $m^{'}$. Elle est calculée comme une moyenne de matrices de covariances de composantes gaussiennes f_{ip}, en se basant d'abord sur m puis sur $m^{'}$. Par exemple, dans la figure (3.9), f_1 et f_2 sont attachées à g_1 selon m. Puis selon $m^{'}$, la covariance $\Sigma_{g_{13}}$ de la composante g_1 est mise à jour à partir de $\Sigma_{f_{23}}$, $\Sigma_{f_{11}}$ et $\Sigma_{f_{12}}$.

Notons que la fonction $m^{'}$ est non surjective, c.à.d qu'une composante $(\mu_{g_j}, \Sigma_{g_{jp}})$ peut n'être attachée à aucune composante de f. Dans ce cas, la matrice de covariance de cette composante ne change pas. Elle pourrait être rattachée à des covariances des composantes f_{il} dans les itérations

Entrées : Deux mélanges f et g de distributions de *Student*, composés respectivement de K et $K^{'}$ composantes $(K > K^{'})$. Les valeurs initiales de moyennes μ_{g_j} sont choisies selon la technique de *K-means* ++, les matrices de covariances Σ_{g_j} sont initialisées à Σ_j/p, où Σ_j est choisie aléatoirement et $(1 < p < P)$.

1 - Approximer chaque composante de *Student* de f et de g par un mélange gaussien de P composantes ;

2 **répéter**

3 - Trouver la meilleure correspondance m, puis calculer l'approximation de la divergence de *KL* entre les composantes de *Student* f_i et $g_{m(i)}$ (approximés par de mélanges gaussiens) de f et g comme suit :

4
$$\begin{aligned} d(f,g) = & \quad d(f,g,m) \\ & = \sum_{i=1}^{K} \pi_{f_i} KL_{match}(f_i||g_{m(i)}) \end{aligned}$$

5 - Mettre à jour les paramètres de modèle g seulement à partir des paramètres de f en se basant sur la fonction de correspondance m :

6
$$\begin{aligned} \mu_{g_j} = & \quad \tfrac{1}{\pi_{g_j}} \textstyle\sum_{i \in m^{-1}(j)} \pi_{f_i} \mu_{f_i} \\ \Sigma_{g_{jp}} = & \quad \tfrac{1}{\pi_{g_j}} \textstyle\sum_{i \in m^{-1}(j)} \pi_{f_i} \\ & \left(\tfrac{1}{|m'^{-1}(p)|} \textstyle\sum_{l \in m'^{-1}(p)} (\Sigma_{f_{il}} + (\mu_{f_i} - \mu_{g_j})(\mu_{f_i} - \mu_{g_j})^T) \right) \end{aligned}$$

7 **jusqu'à** *le critère de convergence de (3.18) n'est pas encore minimisé*;

Sorties : Mélange g, la réduction optimale du f

Algorithme 6: Algorithme de réduction de mélanges de distribution t

restantes.

Rappelons que les résultats de l'algorithme de réduction dépendent directement de l'initialisation du mélange réduit g. Pour cela, nous avons proposé d'utiliser la technique *K-means++* pour initialiser les paramètres de ce modèle [7]. Cette technique, qui a été développée au départ pour choisir les centres initiaux de clusters dans l'algorithme de clustering *K-means* [63], a été adaptée ici pour fonctionner sur les modèles. Les centres choisis dans notre algorithme sont les composantes de *Student* et la distance utilisée est la divergence *KL*. Les étapes principales de l'algorithme *K-means++* sont présentées dans l'algorithme (7).

Nous avons présenté dans cette section, notre méthode pour réduire un mélange de distribution de *Student*. Par la suite, nous allons valider nos propositions avec des expériences : approximation de la distribution de *Student* et réduction d'un mélange de *Student* .

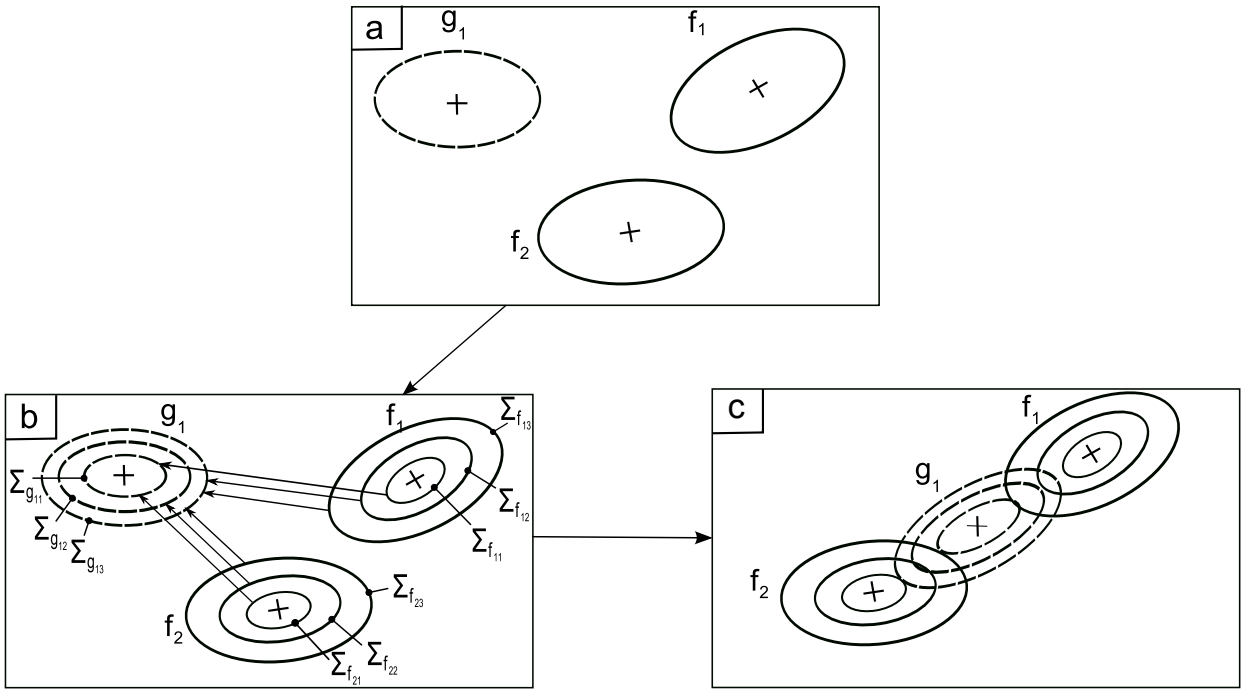


Figure 3.9 – Exemple d'application de l'algorithme de réduction d'un mélange de distribution de *Student*. Les ellipses continues et pointillées représentent respectivement les modèles de f et g. La figure (a) contient les composantes originales de distribution de *Student*. Dans la figure (b), chaque composante *Student* est approximée par un mélange de gaussiennes de $P = 3$ composantes. L'optimisation du critère de *match bound* consiste à chercher la meilleure fonction de correspondance m entre les composantes de f et g, d'une façon à minimiser la sommation de la divergence de *KL*. Ici, les flèches montrent la fonction de correspondance $m^{'}$ obtenue implicitement par la méthode *match bound* entre les composantes f_{ip} et g_{jp}, après avoir lié les composantes f_i avec les composantes g_i selon m. La figure (c) représente la mise à jour de composantes du modèle réduit g. Les paramètres du modèle g sont calculés à partir de ceux du modèle f selon la fonction m.

3.6 Expériences

Dans cette section, la technique proposée pour approximer la distribution de *Student* par un mélange gaussien, ainsi que l'algorithme de réduction d'un mélange de distribution de *Student* vont être évalués. Tout d'abord nous allons montrer comment le nombre de composantes gaussiennes, nécessaires pour obtenir des bonnes approximations, varie en fonction du degré de liberté. Puis dans un deuxième temps, nous allons voir comment un modèle de mélange de *Student* peut être réduit en un autre mélange avec un nombre inférieur de composantes.

3.6.1 Approximation de la distribution de *Student* par un mélange fini de gaussiennes

Le but de cette expérience est de trouver la meilleure approximation d'une distribution de *Student* par une distribution gaussienne. Nous avons vu que cette approximation varie en fonction de degré de liberté

Entrées : Ensemble de composantes de distribution de *Student* de modèles à agréger
1 - choisir un centre aléatoirement parmi l'ensemble de composantes
2 **répéter**
  – pour chaque composante i, calculer $Div(i)$, la divergence *KL* entre i et le plus proche centre qui a été déjà choisi
  – sélectionne un nouveau centre aléatoirement, avec une probabilité proportionnelle au $Div^2(i)$
3 **jusqu'à** P *centres sont choisis*;

Algorithme 7: Algorithme de *K-means++*

ν. Nous avons donc testé pour chaque valeur de ν, le nombre de composantes gaussiennes requis pour avoir une bonne approximation. Pour évaluer les résultats, la divergence de *KL* de *Monte carlo* entre la distribution de *Student* et l'approximation proposée, a été calculée pour chaque nombre de composantes.

Cette expérience est répétée pour tous les degrés de liberté ν entre 1 et 30, en variant chaque fois le nombre de composantes gaussiennes entre 1 et 75. Rappelons que le degré de liberté peut être considéré comme un paramètre important pour préciser le niveau de robustesse des modèles. Ainsi, si le degré de liberté d'une distribution de *Student* est supérieur à 30, il converge vers la loi gaussienne. C'est pour cela, que l'expérience est effectuée seulement pour des valeurs de $\nu < 30$, à partir desquelles la loi de *Student* peut être approximée par une seule gaussienne.

Pour valider notre approximation, nous commençons d'abord par générer, pour chaque degré de liberté, un échantillon de points, à partir de la distribution de *Student* des paramètres (μ, Σ, ν). Nous étudions ensuite l'évolution de l'approximation (pour le même degré de liberté) en fonction du nombre de composantes gaussiennes P. Rappelons que les composantes gaussiennes de cette approximation sont initialisées par la même moyenne μ, mais avec des matrices de covariances différentes. Ces matrices Σ/w sont pondérées par des points générés à partir de la loi *Gamma*.

L'algorithme (8) explique les différentes étapes de cette expérience. Nous l'avons appliqué pour un grand nombre de degré de liberté ν. La figure (3.10) illustre un exemple de plusieurs valeurs de ν. Cette figure montre la variation de *KL* en fonction du nombre de composantes et du degré de liberté. Si le degré de liberté augmente, le nombre de composantes gaussiennes nécessaires pour obtenir un *KL* faible diminue. Remarquons aussi, que pour $\nu \geq 2$, les courbes associées convergent rapidement vers 0 et donnent de bonnes approximations pour des valeurs de P variant entre 8 et 20 composantes. Cela est expliqué par le fait que la distribution de *Student* converge vers une loi gaussienne lorsque $\nu \to \infty$. Cependant pour des petites valeurs de ν, un nombre supérieur à 50 est requis pour réaliser une bonne approximation.

Par la suite, les composantes de *Student* avec un degré de liberté supérieur à 2 sont approximées par 10 gaussiennes et sinon par 50. Ces valeurs ont été déterminées expérimentalement et correspondent aux valeurs à partir desquelles la divergence *KL*, entre la fonction réelle de la distribution de *Student* et son approximation, se stabilise. L'ajout de gaussienne supplémentaire n'accroît pas sensiblement la qualité de l'approximation.

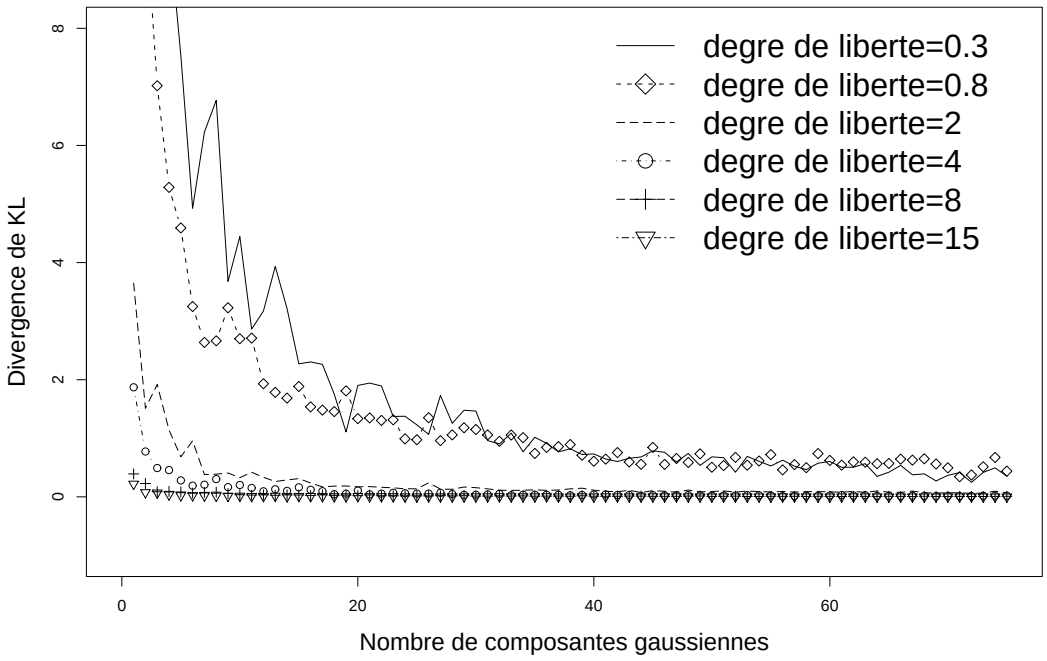


Figure 3.10 – Cette figure montre la variation de la divergence *KL* entre notre approximation et la vraie distribution de *Student*, en fonction du nombre de composantes gaussiennes P et du degré de liberté ν. Elle confirme bien que si le degré de liberté ν augmente, le nombre de composantes nécessaires pour obtenir une petite valeur de *KL* diminue.

3.6.2 Réduction de mélanges de distributions de *Student*

La deuxième expérience évalue notre algorithme de réduction d'un mélange de distribution de *Student*. Nous avons généré pour cela, un mélange f avec des composantes plus ou moins proches entre elles. Le but est de le réduire en un autre mélange g, en regroupant les composantes les plus proches. Le nombre de composantes de modèle g est inconnu. Pour cette raison, l'algorithme de réduction a été lancé plusieurs fois avec un nombre différent de composantes, afin de trouver le nombre correct. Le mélange g minimisant le critère de l'équation (4) a été choisi comme le meilleur mélange réduit.

Nous avons généré aussi un échantillon de données à partir du modèle initial f. Il sert juste à montrer la distribution des données, par rapport au modèle qui les représente (f dans ce cas). Par la suite, nous allons expliquer comment les deux mélanges f et g ont été initialisés.

Initialisation de mélange initial f :

Entrées : 5000 points générés de distribution de*Student* de paramètres (μ, Σ, ν)
Sorties : Nombre de composantes gaussiennes

pour *Chaque degré de liberté* **faire**
  pour $1 \leq P \leq 75$ **faire**
    pour $i = 1$ *à* 20 **faire**
      - Générer P valeurs w_p de la distribution *Gamma*($\nu/2$,$\nu/2$), où ν est le même degré de liberté utilisé dans la distribution de *Student*. Puis initialiser les P composantes gaussiennes avec la même moyenne μ, mais chacune avec la matrice de covariance Σ pondérée par une de valeurs w_p.
      - Calculer la divergence *KL* entre la distribution de *Student* et notre approximation en utilisant la méthode de *Monte Carlo* (équation (3.14) appliquée sur les données en entrée).
    fin
    - Calculer la moyenne de valeurs de *KL* obtenues durant les 20 itérations.
  fin
fin

Algorithme 8: Algorithme de réduction d'un mélange de distribution de *Student*

– les moyennes et les matrices de covariances des composantes initiales du mélange f sont choisies aléatoirement, mais à condition que les matrices de covariance soient inversibles ou régulières. Le fait de choisir des matrices inversibles, est dû à leur utilisation dans le calcul de l'équation (3.13), qui nécessite de les inverser pour calculer la divergence entre les composantes gaussiennes. Notons que si Σ_i (matrice de covariance de composante de *Student* f_i) est inversible, alors $\Sigma_i/w(p)$ (matrice de composante gaussienne associée f_ip) est aussi inversible ;

– les degrés de liberté des composantes de *Student* sont générés selon la loi uniforme avec des valeurs positives. Ensuite, chaque composante de distribution de *Student* est approximée par un mélange gaussien selon la première expérience. D'après les résultats précédents, si le degré de liberté d'une composante f_i est inférieur à 2, alors f_i est représentée par $P_i = 50$ gaussiennes. Sinon, elle est approximée par $P_i = 10$;

– pour chaque composante de distribution de *Student*, sa moyenne est affectée à ses composantes gaussiennes. Rappelons que si Σ_i est sa matrice de covariance, alors les matrices de covariances de ses composantes gaussiennes sont initialisées à $\Sigma_{ip} = \Sigma_i/w(p)$, où $w(p)$ est une valeur d'une variable aléatoire générée suivant la loi *Gamma* $G(\nu/2, \nu/2)$;

Initialisation de mélange réduit g :

– pour chaque composante g_j, le nombre P de ses composantes gaussiennes est choisi comme le plus grand parmi les composantes f_i liées à g_j, selon la correspondance m (voir la section 3.5). Par exemple, si g_j est attaché à f_1 et f_2 selon m, dans laquelle ils possèdent respectivement 10 et 50 composantes, alors le nombre de composantes gaussiennes de g_j est initialisé à 50 ;

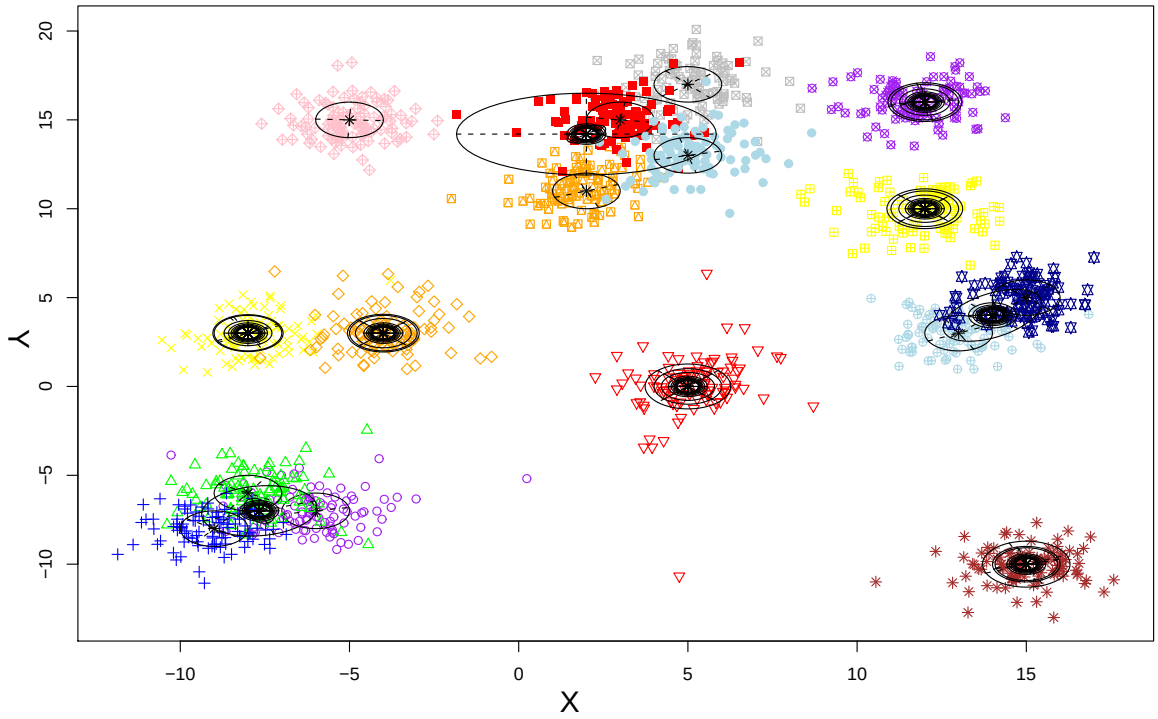


Figure 3.11 – Cette figure présente le modèle f et le modèle réduit g obtenu. Elle illustre aussi les données générées à partir de chaque composante f_i de f. Les ellipses uniques sont les composantes f_i de f, et les ellipses multiples sont les composantes g_j de g.

– les moyennes des composantes de *Student* g_j de g sont choisies selon la technique de *K-means* ++. Rappelons que, les moyennes des composantes gaussiennes g_{jp} sont les mêmes que celle de la composante de *Student* g_j associée ;

– les matrices de covariances des composantes gaussiennes g_{jp} sont initialisées par Σ_j/p, où Σ_j est choisie aléatoirement d'une façon à être inversible et p varie de 1 à P ;

Dans notre expérience, le modèle de f est composé de 17 composantes de distribution de *Student* (voir la figure 3.11). Remarquons qu'il existe des composantes qui sont très proches entre elles (par exemple celles situées entre $(-10, -5)$ selon l'axe X et Y) et d'autres qui sont éloignées entre elles (composante entre $(11, 17)$ selon X et $(-2, -7)$ selon Y).

Pour déterminer le nombre correct des composantes du modèle réduit g, l'algorithme a été lancé plusieurs fois avec un nombre différent des composantes (entre 1 et 17). Le meilleur modèle a été obtenu selon le critère de l'équation (3.16). Dans notre exemple, le mélange réduit g avec 9 composantes a pro-

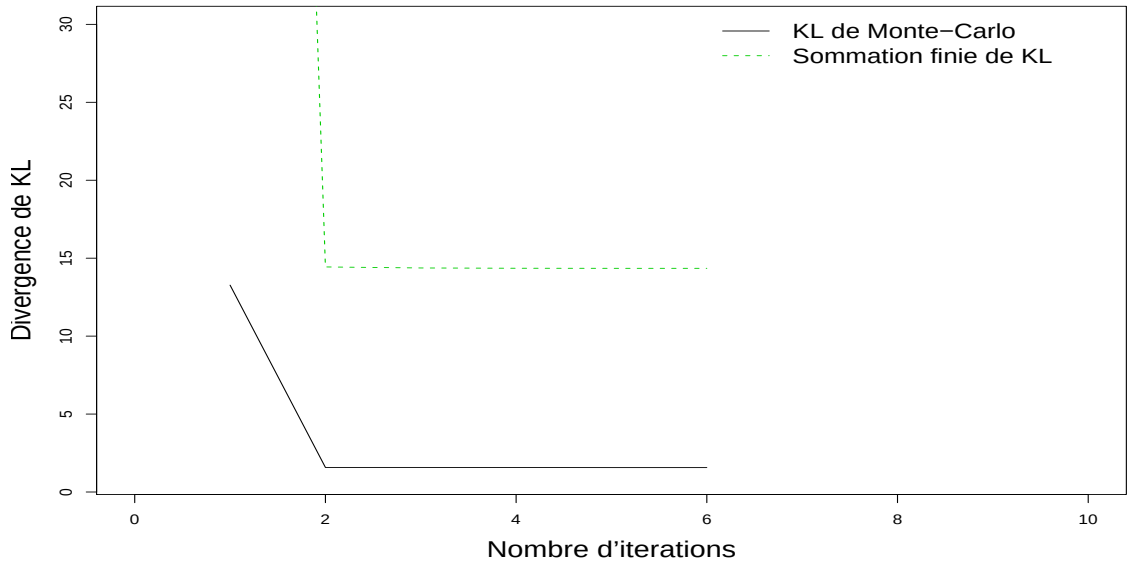


Figure 3.12 – Cette figure montre l'évolution de la divergence de *KL*, le critère d'optimisation de notre algorithme de réduction, tout au long des différentes itérations. Elle mesure à la fin de chaque itération, la divergence *KL* entre le mélange initial f et le mélange réduit g composé de 9 composantes. Cela est calculée par deux méthodes : la méthode de *match bound* choisie comme une méthode de base dans notre thèse, et la méthode de *Monte Carlo* connue par la qualité de ses résultats.

duit la meilleure réduction (valeur minimal de KL_{match}) du mélange f.

La figure (3.11) illustre le modèle initial (ellipses simples) avec les données associées à chaque composante. Elle montre aussi le modèle final obtenu après 6 itérations. Chaque composante de ce modèle est représentée par plusieurs ellipses de même centre. Remarquons que la plupart des composantes de f génèrent une seule composante de g, à l'exception de la composante de f située en $(-5, 15)$, qui est associée à la composante large de g en $(2, 14)$. Les composantes proches sont réduites à une seule composante en g. Par exemple, les 3 composantes situées entre $(-10, -5)$ selon l'axe de X et $(-10, -5)$ selon l'axe de Y sont réduites à une seule composante. Nous remarquons la même chose pour les composantes entre $(0, 7)$ et $(10, 17)$.

La figure (3.12) montre la variation de la divergence de *KL* entre le mélange original f et le mélange réduit optimal g (9 composantes) tout au long des itérations de notre algorithme. Deux méthodes sont utilisées pour mesurer la divergence *KL* : la méthode de *match bound*, qui est caractérisée par un coût de calcul faible, et la méthode de *Monte Carlo* qui réalise la meilleure approximation de *KL*. Remarquons que le critère d'optimisation est réalisé rapidement : la première itération conduit à des moyennes stables pour toutes les composantes gaussiennes. Seules les matrices de covariances changent durant les itérations restantes. Cela peut être expliqué par le fait que les composantes de g se positionnent rapidement

par rapport aux composantes les plus proches de f.

Dans cette expérience, nous avons testé notre algorithme de réduction de mélanges de *Student* dans laquelle les composantes distinctes restent inchangées, tandis que les composantes chevauchées sont regroupées en une seule composante dans g. Selon notre opinion, le mélange g obtenu présente une réduction pertinente du modèle initial f.

3.6.3 Conclusion

Nous avons proposé dans ce chapitre, une méthode d'agrégation des modèles de mélange de *Student*. C'est une technique qui sert à réduire un modèle de mélange de *Student*. Pour cela, nous avons montré dans un premier temps, que la densité de la distribution de *Student* peut être approximée par une somme finie de gaussiennes. Ensuite, nous avons adapté les travaux de *Goldberger et al.* [47] pour qu'ils puissent être appliqués sur le mélange de *Student*.

La technique proposée est specifiée par plusieurs avantages :

– elle est caractérisée par une complexité de calcul faible. Celle-ci s'exprime par le fait que la réduction des mélanges se base seulement sur les paramètres.

– elle permet de réduire un modèle robuste de mélange de *Student*. Autrement dit, c'est grâce à l'utilisation de la distribution de *Student*, qui est connue pour sa robustesse, que notre méthode est considérée pertinente pour résoudre le problème des données atypiques ;

– elle peut être utile pour les applications distribuées, surtout en terme de coût de transmission (seulement les paramètres peuvent être communiqués) ;

Après avoir présenté une technique d'agrégation des modèles de mélange, nous allons montrer dans le chapitre suivant, un état de l'art sur la technique de clustering distribuée. Le but est de montrer les différentes techniques étudiées dans la littérature, afin de positionner notre technique distribuée, qui sera proposée plus tard dans cette thèse.

CHAPITRE 4

État de l'art sur les méthodes de clustering distribuées

4.1 Introduction

Dans ce chapitre, nous présentons un état de l'art sur les techniques de clustering distribuées. Nous commençons d'abord par décrire en général les étapes principales de ces techniques. Ensuite, nous montrons comment elles peuvent être classées selon leurs aspects principaux, en détaillant pour chaque catégorie des algorithmes, tous les choix étudiés dans la littérature. Nous abordons ainsi le problème de données atypiques dans les applications distribuées.

Nous avons vu dans le chapitre (2), que les techniques de clustering classiques ont été étudiées de manière centralisée (toutes les données sont supposées être localisées sur un seul site). Autrement dit, elles ne sont donc pas applicables directement sur des données distribuées, c'est-à-dire réparties sur plusieurs sites. Ainsi, avec l'augmentation continue des données dans le monde réel, l'exploitation centralisée des données est devenue de plus en plus difficile. Celle-ci est due aux coûts de transmission et de calcul qui sont élevés au cas où les données stockées sur chaque site sont très volumineuses, ou encore pour des raisons de respect d'anonymat [80]. D'où la nécessité parfois d'effectuer un clustering distribué sur les données réparties sans les transmettre vers un site central.

Notons d'abord que les méthodes de clustering distribuées peuvent être appliquées de plusieurs façons dans les environnements distribués :

- *algorithmes centralisés/données centralisées :* c'est le type standard des applications dans laquelle la méthode de clustering et les données se trouvent sur une même machine [12, 121] ;

- *algorithmes décentralisés/données centralisées :* dans ce cas, les données sont localisées sur une même machine et les algorithmes de clustering sont exécutés sur plusieurs machines. Notons que c'est le cas du calcul parallèle [29, 123] ;

- *algorithmes centralisés/données décentralisées :* les données sont dispersées sur plusieurs machines ou sites, et l'algorithme de clustering s'exécute uniquement sur un seul site. Les applica-

tions *Web* ont un exemple de ce type de méthodes ;

– *algorithmes décentralisés/données décentralisées :* les données et les algorithmes de clustering sont décentralisés [23, 30, 65].

Notons que notre approche est appliquée d'une manière décentralisée sur des données supposées être réparties sur plusieurs sites. Elle se situe donc dans la quatrième famille des méthodes (algorithmes décentralisés/données décentralisées) citées ci-dessus.

Dans la suite, nous allons montrer tout d'abord l'aspect général des techniques de clustering. Nous proposerons ensuite plusieurs taxonomies de méthodes existantes, en regroupant chaque fois celles qui ont les mêmes propriétés. Ainsi, nous allons positionner en même temps notre technique de clustering, qui sera détaillée dans le chapitre suivant, par rapport aux taxonomies obtenues. Puis, le problème de présence des données atypiques dans les méthodes de clustering distribuées va être étudié. Nous présentons pour cela, plusieurs techniques pour détecter ce type de données. Finalement, nous allons expliquer le protocole de *gossip* qui sera utilisé comme protocole de communication dans notre méthode de clustering proposée.

4.2 Algorithme de clustering distribué

Dans cette section, nous montrons tout d'abord l'approche générale des techniques de clustering distribuées. Nous proposons ensuite de regrouper ces techniques selon plusieurs catégories afin de les détailler plus tard dans le chapitre. Puis nous nous concentrons sur les clusterings basés sur les modèles probabilistes.

4.2.1 Approche générale

D'abord, rappelons que l'approche générale des méthodes de clustering distribuées consiste en trois étapes principales :

– un modèle local est calculé sur chaque nœud ;

– les modèles locaux sont agrégés soit par un nœud central, soit par un super-pair dans le réseau P2P, où tous les nœuds réalisent l'agrégation sur les modèles de leurs voisins ;

– le modèle global est calculé ou les modèles locaux sont optimisés à partir de modèles agrégés en deuxième étape.

Il faut juste rappeler avant d'évaluer les méthodes de clustering distribuées, que les méthodes de l'ensemble de clustering peuvent être appliquées facilement dans les environnements distribués. Ces méthodes combinent plusieurs partitions (clustering) effectuées sur un même ensemble de données de mêmes attribut (données homogènes) ou des attributs différents (données hétérogènes). Le but de ces méthodes est d'améliorer la qualité du clustering [40, 108]. Elles se composent en deux étapes princi-

pales qui peuvent être adaptées facilement aux contextes distribués :

– les différentes partitions obtenues dans la première étape de l'ensemble du clustering, peuvent être considérées comme étant des clusteringd obtenus localement sur les différents sites dans le clustering distribué ;

– une fois que les clusterings sont réalisés dans la première étape, la même fonction de consensus utilisée dans l'ensemble de clustering pour combiner les clustering, peut être utilisée dans le clustering distribué pour obtenir le clustering global. Il suffit donc de transmettre seulement les résultats de clustering obtenus localement vers un site central afin d'appliquer la fonction de consensus.

Dans la section suivante, nous allons présenter les aspects principaux selon lesquelles les méthodes de clustering peuvent être classées.

4.2.2 Différents aspects de méthodes de clustering distribuées

Toutes les méthodes de clustering ont le même objectif de regrouper l'ensemble de données similaires. Cependant, elles peuvent le réaliser d'une façon différente. Le but de cette section est de présenter une vue globale des techniques de clustering existantes. Pour cela, nous allons diviser ces méthodes en plusieurs catégories selon les aspects suivants : structure du réseau, données traitées, algorithme distribué. En se basant sur les partitions obtenues, il sera facile de positionner un nouvel algorithme de clustering par rapport aux algorithmes existants. Dans la suite, les méthodes de clustering distribuées vont être classées selon les concepts suivants :

1. **Structure du réseau :** cet aspect est défini pour décrire le modèle du réseau (rôle des nœuds, communication). Nous étudions donc les propriétés suivantes :

   – environnement : le réseau dans lequel l'algorithme est appliqué : LAN, WAN, pair-à-pair, réseau des capteurs ;

   – architecture : pair-à-pair ou maître/esclaves ;

   – communication entre les nœuds : probabiliste ou déterministe.

2. **Données traitées :** elles forment une partie essentielle dans les algorithmes de clustering. Nous nous intéressons aux propriétés suivantes :

   – type de données traitées : homogènes ou hétérogènes ;

   – données communiquées : données réelles, représentatives, ou prototypes.

3. **Algorithme distribué :** ce concept sert à étudier les différents choix des algorithmes qui peuvent être appliqués, ainsi que les modèles que nous souhaitons obtenir (modèle global, optimisation des modèles locaux). Nous nous concentrons donc sur les propriétés suivantes :

– génération des modèles locaux (algorithme appliqué) : hiérarchique, basé sur la densité des points, probabiliste, etc ;

– objectif de l'algorithme : réalisation d'un modèle global, ou optimisation des modèles locaux ;

– cycle de communication : un, deux, ou plusieurs cycles.

Après avoir défini les concepts des méthodes de clustering distribuées d'une manière générale, nous détaillons par la suite ces concepts en présentant pour chacun d'eux des exemples d'application.

4.3 Présentation des algorithmes distribués

L'objectif de cette section est de décrire les différentes propriétés de chaque concept en regroupant en même temps les méthodes possédant la même propriété.

4.3.1 Structure du réseau

Dans cette section, nous nous concentrons sur la structure de réseau en répondant aux questions suivantes :

– dans quel environnement l'algorithme de clustering est-il appliqué : réseau LAN, WAN, pair-à-pair, réseau des capteurs ?

– quelle architecture est utilisée : pair-à-pair ou maître/esclaves, et quel est le rôle de chaque nœud dans chaque architecture ?

– la communication entre les nœuds du réseau est-elle réalisée d'une façon probabiliste ou déterministe ?

Dans la section suivante, nous montrons les environnements utilisés pour réaliser le clustering distribué.

Types des environnements distribués

Les méthodes de clustering proposées dans la littérature ont été appliquées dans des environnements différents. Bien que la plupart des anciennes méthodes sont testées sur des environnements avec des ordinateurs supposés être localisés dans la même zone géographique (LAN, multiprocesseurs, etc), les méthodes récentes sont appliquées plutôt sur des réseaux comme pair-à-pair, réseau de capteurs. Ces derniers environnements sont plus réalistes que les autres, ils correspondent bien à la distribution de données réelles. Au contraire, l'ancien type d'environnement (LAN, multiprocesseurs, etc) impose l'existence des données dans un seul site. Plusieurs types d'environnements peuvent être distingués :

– le premier type d'environnement peut être un ordinateur de plusieurs unités de calcul (multiprocesseurs) avec une mémoire partagée, ou un ensemble de processeurs avec des mémoires distribuées. Ils fonctionnent en parallèle pour réaliser la tâche de clustering (nommé aussi calcul parallèle). Ils peuvent communiquer à travers un protocole d'échange dit *passage de messages*, comme par exemple la norme *MPI* (Message Passing Interface) [29] ;

– un réseau local *LAN* de plusieurs ordinateurs avec une vitesse de transfert de données très élevée [125] ;

– un réseau *WAN*, composé de plusieurs ordinateurs qui peuvent être dans des zones géographiques différentes [103, 65] ;

– un réseau P2P(pair-à-pair) construit sur l'internet [23, 30] ;

– le réseau de capteurs ou réseau sans fil [35, 124, 88].

Nous avons montré plusieurs environnements dans lesquels un algorithme de clustering peut être testé. Dans la section suivante, nous allons voir quelles sont les architectures utilisées dans les méthodes distribuées en précisant en particulier le rôle des nœuds.

Architecture de réseau pour les méthodes de clustering

Il existe deux architectures principales pour les méthodes de clustering distribuées : *maître/esclaves* et *P2P* [54]. Dans l'architecture *maître/esclaves*, un nœud joue le rôle d'un maître, et tous les autres nœuds sont considérés comme des esclaves. Le *maître* est le responsable de la répartition des tâches entre les *esclaves* et de l'agrégation des modèles locaux. Dans l'architecture *P2P*, tous les nœuds réalisent la même tâche et ont la même responsabilité. Ils échangent itérativement entre eux les modèles locaux afin de réaliser le modèle global [23, 30, 29]. Cette architecture a été de plus en plus utilisée récemment due à ses nombreux avantages. C'est pour cette raison qu'elle était choisie comme l'architecture de base dans notre algorithme de clustering. Dans la suite, les deux architectures vont être décrites, ainsi que leurs avantages et leurs inconvénients.

– *maître/esclaves* : l'architecture *maître/esclaves* est formée par un site central (*maître*), dans laquelle son rôle est de contrôler les autres sites (*esclaves*). Dans les méthodes de clustering qui utilisent cette architecture, le nœud *maître* est le responsable du calcul du modèle global. Pour cela, il collecte tous les modèles locaux des autres nœuds (*esclaves*) afin de construire ce modèle [103, 80]. Puis il le renvoie vers les nœuds *esclaves* pour qu'ils mettent à jour leurs modèles locaux, si nécessaire. L'avantage principal de cette architecture est sa simplicité. Cela peut être exprimé par le fait, que tous les modèles sont collectés sur un seul site. Par conséquent, cela facilite leur analyse et leur traitement. Aussi, cette architecture est caractérisée par un coût de communication beaucoup moins élevé que l'architecture *P2P*. Elle nécessite soit un seul cycle (agrégation de modèle global), soit deux au maximum c'est-à-dire que l'agrégation des modèles est suivi par une étape supplémentaire (optimisation de modèles locaux à partir du modèle global agrégé). Cependant, cette architecture a un inconvénient majeur dans le cas où le *maître* tombe en panne ou quitte

le réseau ;

– pair-à-pair (*P2P*) : cette architecture a été développée pour dépasser les problèmes de l'architecture *maître/esclaves*. Dans cette architecture, il n'existe pas de nœud central, et chaque nœud joue à la fois le rôle de *maître* et *esclave*. Les nœuds communiquent directement avec l'ensemble de leurs voisins, et peuvent échanger ainsi entre eux n'importe quelle information. Deux types de l'architecture *P2P* existent : structuré ou non structuré. L'architecture de *P2P* structuré peut être vue comme un compromis entre l'architecture *maître/esclaves* et le *P2P* non structuré. Elle utilise la notion de super pair dans laquelle chaque super pair est un responsable d'un groupe de nœuds. Un nœud dans le réseau ne peut communiquer qu'avec les nœuds qui ont le même super pair, et les super pairs communiquent seulement entre eux [73, 54]. Dans cette architecture, le modèle global est réalisé d'une façon hiérarchique. Les modèles du niveau inférieur sont agrégés pour former les modèles du niveau supérieur jusqu'à la racine [56, 52].

  L'architecture *P2P* est caractérisée par plusieurs avantages comme la dynamicité dans laquelle les nœuds peuvent joindre et quitter le réseau, la fiabilité (la défaillance est manipulée facilement), scalabilité.

Nous avons décrit dans cette section le rôle des nœuds dans les deux architectures existantes : *maître/esclaves* et pair-à-pair. Nous expliquons ensuite comment les nœuds peuvent communiquer entre eux.

Communication déterministe ou probabiliste

Il existe deux manières pour échanger des informations (données, modèles, ..) entre les nœuds : déterministe ou probabiliste. Les nœuds dans les méthodes qui utilisent la communication probabiliste, choisissent aléatoirement d'autres nœuds pour échanger les modèles locaux, tandis que les nœuds dans les méthodes déterministes peuvent contacter tous leurs voisins.

– communication probabiliste : c'est n'est autre que le style de *gossip* (voir la section (4.7)). Dans ce type de communication, chaque nœud choisit aléatoirement un autre nœud parmi ses nœuds voisins, puis chacun d'eux met à jour son modèle à partir de l'union de deux modèles [115, 77]. Les algorithmes qui utilisent ce type de communication convergent exponentiellement vers les résultats corrects [13] ;

– communication déterministe : dans les méthodes qui utilisent la communication déterministe, chaque nœud communique et échange les informations avec tous ses nœuds voisins. Il met à jour son modèle à partir de l'union de son modèle actuel et les modèles de tous ses voisins [23, 30]. Ce type de méthodes converge vers les résultats corrects d'une façon asymptotique [13].

Dans notre méthode proposée, nous avons utilisé un mélange de deux types. D'abord, la topologie de réseau est changée aléatoirement, c'est-à-dire chaque nœud, durant chaque itération, peut contacter des nouveaux nœuds voisins. Puis, chaque nœud met à jour son modèle à partir de tous les modèles locaux de ses voisins.

Nous avons montré dans cette section les deux types de communication (déterministe, probabiliste). Cependant, il existe parfois des contraintes sur ces communications. Nous allons présenter quelques unes dans la suite.

Contrainte de communication

La communication dans les algorithmes de clustering dépend parfois de plusieurs facteurs : la nature des données ou l'environnement du réseau dans lequel l'algorithme est appliqué.

- Coût de communication : il est parfois difficile de transmettre une grande quantité dans le réseau comme par exemple dans le cas des réseaux de capteurs. Celle-ci est exprimée par le fait que les capteurs ont une durée de vie limitée liée à leurs batteries, et des capacités de stockage et de calculs faibles. Il est donc préférable d'utiliser pour ce type de réseau des techniques de clustering communiquant soit un nombre très petit de données, soit des modèles ;

- Données privées : le problème de confidentialité des données peut être considéré aussi comme une autre contrainte de communication. Les sources ne peuvent pas partager leurs données entre elles. En revanche, elles peuvent communiquer soit des modèles, soit des résumés de modèles.

Après avoir présenté le premier concept utilisé (structure de réseau) pour classer les méthodes de clustering distribuées, nous allons montrer comment elles peuvent aussi être regroupées selon les données traitées.

4.3.2 Données traitées

Dans cette section, nous nous basons sur l'aspect de données étudiées et communiquées pour distinguer les différents algorithmes. Il faut donc répondre aux questions suivantes :

- est ce que les données traitées sont des données homogènes ou hétérogènes ?

- quel est le type de données communiquées ?

Dans la suite, nous allons décrire les catégories de données qui peuvent être étudiées.

Données homogènes ou hétérogènes

Il existe deux types de données qui sont traitées par les méthodes de clustering : les données homogènes et les données hétérogènes. Les méthodes de clustering peuvent être nommées homogènes ou hétérogènes selon le type de données étudiées. Dans notre méthode, nous testons les données homogènes plutôt que celles hétérogènes. Pour distinguer ces deux types, supposons l'existence d'une table qui contient l'ensemble des données, dans laquelle les colonnes représentent les attributs de données, tandis que les lignes correspondent aux enregistrements.

– méthodes homogènes : dans les méthodes de clustering homogènes, la table des données est divisée horizontalement en plusieurs partitions, chaque partition forme l'ensemble des données d'un seul site. Les partitions homogènes contiennent les mêmes colonnes (attributs) mais avec des enregistrement différents. Ce type de données est beaucoup plus étudié que l'autre type hétérogène [29, 123, 30].

– méthodes hétérogènes : dans ce type de méthode, une division verticale de la table des données est effectuée. Les partitions hétérogènes contiennent les mêmes enregistrements mais chacun d'eux avec une partie des attributs [69, 71, 112].

Nous avons présenté les types de données traitées. Dans la suite, nous allons montrer les différentes catégories de données communiquées entre les nœuds.

Types de données communiquées

Les méthodes de clustering distribuées peuvent aussi être caractérisées par la quantité de données échangée entre les nœuds. La plupart des méthodes de clustering distribuées cherchent à réduire le coût de communication, en transmettant des représentations synthétiques qui sont beaucoup moins volumineuses que l'ensemble des données traitées. Les méthodes de clustering peuvent être divisées en trois classes selon le type de données communiquées entre les nœuds (ordre croissant en fonction du coût de transmission) :

– communication de prototypes de clusters : ce genre de méthodes est le meilleur en terme de coût de transmissions, dont un nombre très petit de prototypes représentatifs est communiqué entre le nœuds. Les prototypes sont calculés synthétiquement à partir des données, ils sont utilisés comme des représentants de clusters. Les prototypes peuvent prendre plusieurs formes, soit des centroïdes de clusters comme dans [23, 30], soit des dendrogrammes [103], soit des modèles génératifs [80] ;

– communication de données représentatives : une autre façon de réduire le coût de communication consiste à échanger un nombre limité de données représentatives. Pour cela, chaque nœud choisit parmi l'ensemble de ses données locales, celles qui se trouvent dans des régions denses. Elles sont moins nombreuses par rapport à l'ensemble des données traitées. Un exemple d'application est présenté dans [65]. Les méthodes qui communiquent les données représentatives limitent donc la quantité de données transmises ;

– communication de données : elle consiste à échanger les données actuelles entre les nœuds. Par exemple, le cas le plus simple consiste à transférer toutes les données distribuées vers un nœud central pour un clustering global. Un autre exemple qui utilise ce type de communication a été proposé dans [55] dans laquelle les nœuds échangent entre eux les données afin d'optimiser les clustering locaux. La communication de données est la plus coûteuse parmi les autres types de communication. En plus, l'utilisation de cette technique est limitée due à son inefficacité de résoudre les problèmes principaux dans les systèmes distribués (coût de communication, confidentialité de données).

Nous avons présenté les différentes manières pour classer les techniques de clustering selon les données (traitées, communiquées). Nous décrivons ensuite le concept lié directement aux algorithmes appli-

qués.

4.3.3 Algorithme distribué

Dans cette section, nous cherchons à répondre aux questions suivantes :

- quel algorithme de clustering est appliqué pour générer les modèle locaux ?
- l'objectif principal des algorithmes est-il de calculer un modèle global ou d'améliorer seulement les modèles locaux ?
- combien de cycles de communication a besoin un algorithme pour réaliser le clustering ?

Dans la suite, nous allons voir comment est obtenu le modèle local.

Modèle local

Toutes les méthodes de clustering distribuées commencent par calculer les modèles locaux sur chaque nœud. Cette étape est vue comme une étape classique qui consiste à appliquer d'une façon indépendante sur les données locales un algorithme de clustering. Les modèles locaux obtenus peuvent être générés de plusieurs façons, tout dépend de l'algorithme de clustering appliqué pouvant être soit un algorithme hiérarchique [103, 27, 69], soit un algorithme basé sur les densités de données [65, 123], soit un algorithme basé sur la distribution de données [88, 49, 124, 119, 13, 77, 35], soit un algorithme de partitionnement [29, 23, 30, 56, 52]. Ils peuvent prendre plusieurs formes : des modèles probabilistes comme dans [80], des prototypes comme dans [23, 30], ou des données représentatives comme dans [65].

Après avoir présenté les différentes manières pour générer le modèle local, nous nous intéresserons par la suite sur l'objectif des algorithmes de clustering distribués.

Optimisation du modèle global ou local

Les méthodes de clustering distribuées peuvent se distinguer selon le modèle souhaité optimisé : local ou global. En d'autres termes, après avoir estimé les modèles locaux sur les différents nœuds, les nœuds communiquent entre eux, soit des modèles locaux afin de calculer le modèle global, soit des données actuelles dont le but est d'améliorer les modèles locaux. Dans la suite, les méthodes de clustering peuvent être classées selon le modèle qui visent à l'optimiser :

- optimisation du modèle global : le modèle global, par définition, est le modèle qui représente l'ensemble des données distribuées dans le réseau. Autrement dit, c'est un clustering global construit à partir de l'ensemble des clustering locaux obtenus sur tous les nœuds du réseau. Donc le modèle global forme une connaissance globale commune à tous les nœuds, et chaque nœud peut améliorer la qualité de son clustering local en se basant sur ce modèle. La plupart des méthodes de clustering implémentées cherchent à optimiser le modèle global. Ce modèle global peut être calculé de

plusieurs façons, tout dépend de l'architecture du réseau (P2P, maître/esclaves) utilisée qui joue un rôle principal dans son calcul (voir la section 4.3.1) ;

– optimisation du modèle local : il existe d'autres techniques qui ont pour but d'optimiser les modèles locaux plutôt que le modèle global. Ces techniques n'imposent pas un seul ensemble de clusters pour tous les nœuds, et chaque nœud peut avoir à la fin son propre clustering qui reflète les caractéristiques de ses données locales. Ce type de méthodes a été étudié dans [55]. Les nœuds ici travaillent en collaboration en échangeant entre eux leurs données actuelles pour améliorer la qualité de leurs clusterings locaux.

Les méthodes qui ont pour but d'optimiser le modèle global offrent une vue globale de clustering pour tous les nœuds, sans besoin de transmettre toutes les données vers chaque site. Les méthodes qui visent à optimiser les modèles locaux apparaissent utiles dans le cas où l'ensemble des clusters, sont souhaités être sur un seul nœud et non pas sur tous les nœuds [73].

Dans notre méthode proposée, l'objectif est de calculer le modèle global, pour qu'il soit accessible par tous les nœuds. Pour cela, tous les nœuds échangent entre eux leurs modèles locaux afin de les agréger et de réaliser le même modèle sur tous les nœuds.

Nous avons vu dans cette section comment est réalisé l'objectif des méthodes. Nous montrons ensuite combien de cycles a besoin un algorithme pour converger vers les résultats finaux.

Cycle de communications

Chaque algorithme de clustering nécessite un certain nombre de cycles pour effectuer le clustering (un, deux ou multiple). Nous pouvons donc classer les méthodes selon le nombre de communications requises entre les nœuds.

– un cycle de communication : les méthodes qui utilisent un seul cycle, consiste à envoyer tous les modèles locaux vers un site central, afin de les agréger et calculer le modèle global [103, 80] ;

– deux cycles de communications : d'autres méthodes nécessitent deux cycles de communications pour réaliser le clustering. Elles calculent aussi le modèle global de la même façon que les méthodes d'un seul cycle, mais avec une étape supplémentaire visant à optimiser les modèles locaux à partir du modèle global agrégé [65]. D'où le besoin d'un autre cycle ;

– multiples cycles de communications : les nœuds dans ces méthodes échangent leurs modèles avec leurs voisins d'une façon itérative jusqu'à ce qu'ils atteignent le même modèle global. Pour réaliser ce modèle, chaque nœud a besoin donc de communiquer plusieurs fois ses voisins. Ce type de méthodes est très implémenté [73, 30, 23, 56, 52], particulièrement dans les systèmes purement décentralisés comme le système P2P. Contrairement aux deux premières catégories, les problèmes du nœud central n'apparaît pas (tombe en panne, quitte le réseau). Tous les nœuds fonctionnent en parallèle pour obtenir le clustering final en partageant le calcul entre eux.

Dans notre méthode, les nœuds réalisent le clustering en échangeant d'une façon continue leurs modèles entre eux. Elle utilise donc plusieurs cycles de communications.

Après avoir présenté les différents aspects selon lesquels les méthodes de clustering distribuées peuvent être classées, nous allons détailler dans la section suivante plusieurs techniques basées sur les modèles probabilistes.

4.4 Clustering basé sur les modèles probabilistes

Nous avons vu dans le chapitre (1), que l'un des objectifs principaux de cette thèse est d'utiliser les modèles de mélanges pour réaliser un clustering distribué. Rappelons que ce choix a été motivé pour plusieurs raisons (coûts de transmission et de calcul faibles, problème de confidentialité résolu). Pour cette raison, nous nous concentrons sur cette partie sur le clustering basé sur des modèles probabilistes.

Dans la suite, plusieurs versions distribuées de l'algorithme *EM*, qui est très connu pour l'estimation des modèles probabilistes, vont être détaillées [88, 49, 124, 119, 13, 77, 35]. Ces différentes techniques de *EM* ont le même objectif d'estimer le modèle global dans le réseau, mais la différence réside dans la façon d'obtenir ce modèle. Dans la suite, plusieurs implémentations de l'algorithme *EM* distribué vont être citées.

Rappelons d'abord que l'algorithme *EM* se déroule d'une façon itérative en alternant entre deux étapes principales : E(Expectation) et M(Maximization). Ces deux étapes sont étendues pour s'adapter aux environnements distribuées. L'étape *E* est calculée de la même façon dans tous les algorithmes proposés, et la différence apparaît juste dans la calcul de l'étape *M*. La plupart de méthodes (proposées pour implémenter l'algorithme *EM* dans l'environnement distribuée) se focalisent sur la distribution de l'étape *M*, en particulier sur la façon de communiquer les paramètres de modèle entre les nœuds du réseau, pour estimer les nouveaux modèles.

- étape *E* : l'implémentation distribuée de l'étape *E* est facile, elle ne nécessite pas de communication : étant donné les paramètres de modèle, chaque nœud calcule le degré d'appartenance (probabilité a posteriori) de ses données locales aux différentes composantes de modèles ;

- étape *M* : l'étape M, qui consiste à mettre à jour les paramètres de modèle, est calculée de plusieurs manières dans les algorithmes proposés. Dans [88], les auteurs utilisent une topologie en anneau crée sur le réseau, dans laquelle tous les nœuds sont liés dans une boucle fermée. L'étape *M* est calculée ici d'une façon cyclique. Dans le premier sens, le modèle est mis à jour sur chaque nœud, puis il est transmis vers le successeur jusqu'au dernier nœud dans le chemin. Ce nœud est responsable du calcul du modèle final pour l'itération courante. Ce modèle calculé est envoyé dans le sens inverse à tous les nœuds, afin de lancer une nouvelle itération. Cet algorithme converge plus rapidement que l'algorithme standard *EM* dû à sa façon incrémentale de mettre à jour les paramètres. Mais il ne passe pas à l'échelle à cause de sa topologie en anneau qui nécessite de lier tous les nœuds dans le réseau.

  Dans [49, 124], chaque nœud met à jour son modèle à partir de ses voisins directs. Pour cela, il continue à échanger périodiquement les informations avec eux jusqu'à obtenir le modèle global

final. Ces algorithmes convergent asymptotiquement vers les résultats corrects, mais ils nécessitent des synchronisations entre les nœuds tout au long de leurs exécutions.

L'algorithme *Newscast EM* [77] utilise le protocole de communication probabiliste *gossip* pour mettre à jour les paramètres de modèle dans l'étape *M*. Cet algorithme converge exponentiellement vers les résultats corrects. Mais pratiquement, cette technique nécessite un réseau complètement connecté, pour que chaque nœud puisse contacter n'importe quel autre nœud dans le réseau.

Les deux techniques dans [119, 13] utilisent des arbres comme une architecture entre les nœuds pour mettre à jour les paramètres du modèle global. La méthode dans [13], utilise de plus une phase de surveillance pour vérifier si le modèle est toujours valide pour les données actuelles. Si ce n'est pas le cas, l'algorithme *EM* est appliqué d'une façon centralisée sur un échantillon de données sur la racine de l'arbre. Ces algorithmes ont une complexité de calcul élevée due à la construction de l'arbre imposé.

Notons que notre méthode de clustering proposée (voir le chapitre suivant), commence à générer localement sur les nœuds des modèles probabilistes représentant les données des nœuds. Ces modèles locaux sont estimés une fois par l'algorithme *EM*. Le modèle global est ensuite réalisé d'une façon itérative en combinant dynamiquement les modèles locaux. L'agrégation des modèles est effectuée directement à partir des paramètres de modèles.

Après avoir présenté un état de l'art de la technique de clustering distribuée, nous traiterons ci-dessous le problème de robustesse dans les applications distribuées.

4.5 Estimation robuste distribuée

Nous avons décrit avant une vue globale des méthodes de clustering distribuées. Nous avons montré aussi que ces méthodes peuvent être divisées en plusieurs catégories. Cependant elles n'ont pas pris en compte le problème de présence des données atypiques dans l'ensemble de données étudiées. Il existe deux façons pour ignorer l'influence de ce type de données soit par les méthodes statistiques [109, 107, 81], soit par les techniques de détection des données atypiques [131, 91, 16]. Les méthodes statistiques robustes consistent à estimer les paramètres de modèle en ajustant le maximum la majorité des points. Les données atypiques peuvent être identifiées après selon leurs déviations par rapport au modèle ajusté. En revanche, le but des méthodes de détection des données atypiques est d'identifier ces données, puis de les supprimer et d'analyser les données restantes après, si nécessaire.

Dans la suite, nous allons présenter quelques approches décentralisées statistiques robustes, ainsi que plusieurs techniques décentralisées pour la détection des données atypiques.

4.5.1 Méthodes statistiques robustes

Les méthodes statistiques robustes ont été les premières méthodes développées pour résoudre le problème de données atypiques [101, 99]. Le but principal de ces méthodes est d'estimer un modèle représentant la majorité des points en ignorant l'influence des données atypiques.

Nous avons vu dans le chapitre (2) (voir la section 2.8 plusieurs techniques statistiques robustes [44, 41, 93]. Toutes les méthodes présentées dans cette section sont calculées d'une façon centralisée, c'est-à-dire que toutes les données sont localisées dans un seul endroit. Cependant, plus récemment, la plupart des méthodes robustes proposées ont été développées pour être appliquées dans des environnements distribués. Par exemple, plusieurs approches ont été implémentées pour traiter le problème des informations atypiques (incertitudes dans les mesures, liens défaillants, données, modèles) dans les algorithmes des agrégations distribuées.

Les informations atypiques peuvent varier d'une approche à une autre, par exemple les travaux présentés dans [120] prennent en compte la présence des incertitudes dans les mesures effectuées. Les auteurs dans [92] calculent la fonction de consensus en considérant l'existence de liens défaillants. D'autres travaux, supposent que certains nœuds peuvent jouer le rôle des nœuds malicieux et produire des valeurs fausses [109, 107].

Dans *De-RANSAC*, les nœuds sont supposés maintenir des données atypiques, et que tous les nœuds de réseau travaillent en collaboration dans le but de réaliser une estimation robuste du modèle global. Dans cette méthode, chaque nœud génère plusieurs échantillons de données parmi l'ensemble de ses données et celles de ses voisins. Puis tous les nœuds communiquent ses échantillons afin de trouver le meilleur d'entre eux. Enfin, le modèle global est estimé à partir de l'échantillon obtenu.

Notre approche est similaire à celle de *De-RANSAC*. En comparaison avec *De-RANSAC*, qui calcule directement le modèle à partir des données, notre approche cherche le modèle global à partir de modèles locaux générés localement sur les nœuds (non pas sur les données). Ainsi, chaque nœud dans notre méthode génère plusieurs échantillons à partir de son modèle et les modèles de ses voisins. Puis, il calcule le nouveau modèle local à partir du meilleur échantillon trouvé. Dans *De-RANSAC*, chaque nœud génère de la même façon plusieurs échantillons de données, puis tous les nœuds communiquent leurs échantillons afin d'appliquer une technique de vote pour choisir le meilleur d'entre eux.

Après avoir cité quelques approches statistiques décentralisées pour obtenir des estimations robustes, nous montrons ensuite plusieurs techniques pour la détection des données atypiques, réalisées aussi d'une manière décentralisée.

4.5.2 Détection décentralisée des données atypiques

Ce type de méthodes consiste à identifier les données atypiques d'une manière décentralisée, c'est-à-dire sans besoin de transmettre toutes les données vers un site central (les données sont supposées réparties sur plusieurs sites). Les méthodes décentralisées, utilisent la notion de données atypiques locales et globales. Chaque nœud calcule les données atypiques locales soit à partir de ses données propres [131, 91], soit de l'ensemble de ses données et celles de ses voisins [16]. La plupart de ces méthodes servent à calculer aussi les données atypiques globales en se basant sur les données atypiques obtenues localement sur les nœuds [16]. Dans la suite, nous décrivons plusieurs techniques pour la détection décentralisée de données atypiques.

Les travaux de [131] consistent à trouver les données atypiques globales sur un réseau de données distribuées. D'abord, tous les nœuds calculent les données atypiques dans leurs partitions locales, en

se basant sur les distances entre les points. Ensuite, toutes les données locales sont transmises vers un nœud central, qui est responsable d'identifier les données atypiques globales. Cette méthode est simple à appliquer. Cependant les résultats de cette technique dépendent directement des paramètres définis par l'utilisateur (rayon de voisinage, nombre minimal de points dans le voisinage).

Les auteurs de [91] ont proposé une technique pour calculer les données atypiques globales dans un réseau de capteurs. Ils ont imposé une architecture particulière composée de capteurs de capacités faibles et fortes. Les capteurs faibles détectent les données atypiques locales, tandis que les capteurs des capacités fortes sont les responsables de l'identification des données atypiques globales. Cette méthode utilise la fonction de noyau pour estimer la distribution des données locales. Pour identifier les données atypiques, elle évalue chaque point avec son voisinage. Elle dépend directement des paramètres de l'utilisateur comme le rayon de voisinage. Elle suppose ainsi l'existence d'une architecture particulière. Cependant elle trouve les données atypiques d'une manière incrémentale.

Les travaux de [126] utilisent une topologie hiérarchique entre les nœuds pour identifier les données atypiques. Le but est de réduire le coût de transmission entre les nœuds. La racine de l'arbre hiérarchique calcule les données atypiques globales à partir de celles locales reçues par les autres nœuds dans l'arbre. Puis elle les diffuse vers tous les nœuds. Les nœuds répètent la même opération jusqu'à obtenir les données atypiques finales. Cette méthode impose une topologie hiérarchique. Elle peut être appliquée seulement sur les données unidimensionnelles.

Les auteurs dans [16] réalisent une technique qui trouve les données atypiques globales d'une manière complètement décentralisée. Chaque nœud continue à échanger ses données atypiques locales avec ses voisins immédiats jusqu'à ce qu'aucun changement ne soit réalisé. À ce point, tous les nœuds contiennent les données atypiques globales. Pour garantir la convergence, cette méthode suppose que le réseau reste statique(pas de changement dans le réseau pendant une période de temps définie par l'utilisateur).

Dans notre méthode, nous utilisons une technique pour trouver des modèles non fiables plutôt que les données. Le but est de réaliser une estimation robuste de modèle global représentant l'ensemble des données dans le réseau. Pour cela, chaque nœud cherche à identifier localement les modèles non fiables à partir de l'union de son modèle et celles de ses voisins. Ces modèles sont supprimés et une étape d'agrégation est effectuée sur les modèles restants. Notre technique de détection des modèles non fiables est appliquée d'une manière décentralisée dans laquelle chaque nœud communique seulement ses voisins. Cependant nous ne cherchons pas à calculer les modèles non fiables globaux. Ceci peut entraîner par exemple la déclaration d'un modèle local comme non fiable, sachant qu'il est considéré fiable en global, c'est-à-dire par rapport à l'ensemble des modèles dans le réseau.

Nous avons présenté un état de l'art sur la technique de détection de données atypiques. Dans la suite, nous décrivons le protocole *gossip*, en particulier le *Peer Sampling Service*, utilisé pour changer la topologie de réseau.

4.6 Diffusion de l'information dans le réseau

Nous avons vu avant qu'il existe deux architectures de réseau : *maître/esclaves* et *P2P* (voir la section 4.3.1). Rappelons que grâce à ses nombreux avantages (dynamicité, fiabilité, scalabilité), l'architecture *P2P* a été utilisée de plus en plus récemment. Dans cette architecture, chaque nœud communique directement avec l'ensemble de ses voisins. Nous avons présenté ainsi dans la même section deux manières de communication (pour échanger des informations) entre les nœuds d'un réseau : probabiliste (ou style de *gossip* dans laquelle chaque nœud choisit aléatoirement un autre nœud parmi ses nœuds voisins) et déterministe (chaque nœud se communique avec tous ses nœuds voisins). Rappelons aussi que dans notre méthode, nous avons choisi un mélange de deux techniques (chaque nœud applique un style de *gossip* pour changer la topologie du réseau et communique avec tous ses voisins pour récupérer leurs modèles). Dans la suite, nous décrivons certains exemples d'applications déterministes puis nous détaillons dans la section suivante le protocole de *gossip*.

Exemples de protocoles de communication déterministe
Nous présentons dans cette section plusieurs techniques de communication (déterministe) entre les nœuds d'un réseau [88, 30, 23]. Le but principal de ces travaux est de réaliser un clustering distribué. Mais nous nous intéressons ici à la manière de communication utilisée pour effectuer cette tâche.

Par exemple, les auteurs de [88] utilisent une topologie en anneau (c'est-à-dire que chaque nœud possède un successeur et un prédécesseur) pour effectuer un clustering global. Dans un premier sens, le modèle global est calculé petit à petit jusqu'à arriver au dernier nœud dans la boucle (ce nœud est responsable de la réalisation du modèle final de chaque itération). Le modèle obtenu sur ce dernier nœud est transmis dans le sens inverse vers tous les nœuds afin de lancer une nouvelle itération. Cette technique impose une topologie en anneau qui englobe tous les nœuds du réseau. Elle nécessite donc une synchronisation élevée entre les nœuds. Celle-ci peut causer des problèmes (reconstruction de la topologie du réseau) lorsqu'un changement se produit dans le réseau (nœuds rejoignent ou quittent le réseau ou tombent en panne). Elle n'est pas donc adaptable à grande échelle (c'est-à-dire avec un nombre très grand de nœuds).

Les travaux de [30] cherchent à réaliser un clustering (utilisation de l'algorithme *K-means*) d'un ensemble de documents distribués sur un réseau *P2P*. Pour échanger des informations (prototypes de clusters) entre les nœuds, le mécanisme *Probe/Echo* a été utilisé. Ce mécanisme consiste d'abord à définir un nœud initiateur (responsable du lancement de l'algorithme). Ce nœud envoie un message *Probe* (avec les prototypes) à tous ses voisins. À la réception de ce message, chaque nœud à son tour envoie le message *Probe* à ses voisins (sauf l'expéditeur de *Probe*) jusqu'à ce que tous les nœuds reçoivent ce message. Pour mettre à jour le modèle global, chaque nœud attend un message *Echo*(avec les nouveaux prototypes) de chacun de ses voisins. Si un nœud p a reçu ce message de tous ses voisins, il l'envoie au premier nœud à envoyer *Probe* au p. Cette opération est répétée jusqu'à ce que le nœud initiateur reçoive un *Echo* de tous ses voisins. Cette technique nécessite donc une synchronisation globale complète entre les éléments du réseau. Cependant grâce à cette manière de communication (mécanisme *Probe/Echo*) elle peut fournir des résultats corrects (c'est-à-dire identiques aux résultats obtenus par l'algorithme *K-means* sur des données centralisées).

Une technique similaire qui consiste à appliquer l'algorithme *K-means* sur un réseau *P2P* a été proposée dans [23]. Elle nécessite seulement des communications et des synchronisations locales à chaque itération (chaque nœud communique et synchronise seulement avec ses voisins). L'algorithme est initialisé sur un nœud quelconque p (responsable de générer les informations initiales). Ensuite p envoi ces informations à tous ses voisins qui à leurs tours les transmettent vers leurs voisins et ainsi de suite. Durant chaque itération, chaque nœud envoi un message (nommé un message d'appel) à tous ses voisins. Ce message sert comme une demande à tous les voisins pour qu'ils répondent par leurs informations locales (étape de la mise à jour des informations globales). Ces messages sont enregistrés dans une table sur chaque nœud. Si les informations locales d'un nœud ne changent pas entre deux itérations consécutives (différence est inférieure à un seuil défini par l'utilisateur) alors il entre dans un état terminé. Si tous les nœuds arrivent à cet état alors l'algorithme s'arrête. Cette technique est considérée pertinente pour les réseaux dynamiques (elle n'impose pas la stabilité ni pour la topologie du réseau ni pour les données réparties sur les nœuds).

Après avoir expliquer plusieurs exemples d'applications de communications déterministes, nous allons voir dans la suite le protocole de *gossip*.

4.7 Protocole de *gossip*

Le protocole de *gossip* (ou épidémique), par définition, est la façon probabiliste de choisir deux éléments parmi un ensemble d'éléments pour communiquer et échanger des informations entre eux [34, 75]. Il sert donc à diffuser aléatoirement certaines informations (données, modèles,...) dans un réseau afin de réaliser une tâche donnée. Dans les dernières années, la communication par le protocole de *gossip* dans les systèmes distribués, est devenue un paradigme général pour plusieurs applications, comme la diffusion de l'information, l'agrégation, la gestion de la topologie virtuelle d'un réseau (topologie au dessus du réseau physique), répartition de charge dans les réseaux etc [34, 75]. Une propriété commune à tous les protocoles basés sur *gossip*, est que chaque nœud dans le réseau échange périodiquement des informations avec un autre nœud dans son voisinage. L'importance de ce protocole provient de sa capacité à diffuser l'information entre un grand ensemble de nœuds connectés d'une façon fiable, même si les nœuds joignent ou quittent le réseau fréquemment, et aussi dans le cas de défaillance de liens.

Dans la suite, nous allons expliquer le protocole de *gossip* général qui peut être appliqué dans un réseau donné (P2P, réseaux de capteurs, etc), constitué de plusieurs nœuds interconnectés par des lignes de communication. Puis, nous allons détailler plus tard des implémentations spécifiques de protocole de *gossip* (Peer Sampling Service).

4.7.1 Algorithme général de *gossip*

Considérons tout d'abord l'existence d'un ensemble de nœuds connectés par un réseau. Dans le protocole *gossip*, chaque nœud exécute deux processus séparés : le processus actif et le processus passif. Dans le processus actif, chaque nœud choisit aléatoirement un autre nœud p pour initialiser une communication. Il envoie un message qui contient son état local (information à communiquer) à p, et il attend une réponse de p. Dans le processus passif, chaque nœud se met en attente jusqu'au la réception d'un message demandé par un autre nœud initiateur (état actif), pour lui répondre par son état.

L'algorithme 9 illustre le squelette du protocole de *gossip* général. Ce protocole est décrit ici du point de vue d'un participant *A* qui joue à la fois le rôle d'un nœud actif et passif. D'abord, le nœud *A* exécute la fonction $Choisirnoeud()$ qui consiste à choisir un nœud aléatoirement (supposé *B* dans notre cas), soit à partir du réseau entier, soit de son voisinage. Notons que la sélection à partir du voisinage est plus réaliste. Cela est exprimé par le fait qu'en réalité chaque nœud est connecté seulement à un nombre fini de nœuds. Ensuite, *A* initialise l'échange des états avec *B* en lui envoyant son état, puis il attend sa réponse. Lors de la réception de la réponse de *B*, *A* met à jour son état local selon la méthode $MettreAJour(S_a, S_b)$, qui prend en paramètres les états de *A* et *B*.

Dans l'état passif, le nœud *A* reste en attente jusqu'à la réception d'une demande d'un autre nœud (supposé aussi B). Notons qu'un nœud en état passif, peut répondre à plusieurs nœuds en même temps. Après avoir envoyé son état à *B*, A met à jour son état local aussi selon la méthode $MettreAJour(S_a, S_b)$. Ces opérations entre les nœuds sont répétées périodiquement chaque Δt de temps. Le protocole de *gossip* a été appliqué pour diffuser des informations dans plusieurs applications distribuées [14, 67, 74, 77], dans lesquelles les états de nœuds et la méthode $MettreAJour$ se différent d'une application à une autre. Ils dépendent directement du domaine d'application [34, 75]. Par exemple, dans les applications de gestion de réseaux, l'état d'un nœud A est un ensemble fini des nœuds sélectionnés à partir de son voisinage. La méthode $MettreAJour$ dans cette application, prend en entrée l'union des voisinages du nœud A et ceux reçus de l'autre nœud (B par exemple). Le but est de changer le voisinage de chaque nœud, et par conséquent la topologie de réseau.

Un autre exemple important pour l'application de protocole de *gossip* est dans le domaine de l'agrégation distribuée. Dans cette application, les états des nœuds peuvent être des données ou des modèles de paramètres, etc. La fonction $MettreAJour$ est remplacée dans ce cas par une fonction d'agrégation (par exemple la moyenne mathématique de valeurs des nœuds contactés).

Après avoir présenté le protocole de *gossip*, nous allons montrer dans la section suivante une implémentation de ce protocole, nommé le *Peer Samling Service*.

4.7.2 Peer Sampling Service

Le *Peer Samling Service*, est un service qui consiste à fournir pour chaque nœud une liste de nœuds pour échanger des informations entre eux. En d'autres termes, il sert à changer dynamiquement le voisinage de chaque nœud, et par conséquent la topologie de réseau.

Le *PSS* est une implémentation particulière du protocole de *gossip*. Le nœud A dans ce cas commence par choisir parmi sa liste de voisins un autre nœud B. Puis les deux nœuds sélectionnent une partie de leurs voisinages pour l'envoyer à l'autre. Enfin, chacun d'eux met à jour son voisinage à partir de sa liste courante et celle reçue de l'autre nœud.

Nous avons vu dans la section précédente, que les protocoles basés sur *gossip* ont été largement utilisés dans les applications distribuées comme la diffusion des informations, l'agrégation, la gestion de réseau, etc. Le PSS forme une composante fondamentale pour un grand nombre de ces applications. Celle-ci est exprimée par le fait qu'il fournit périodiquement pour chaque nœud une nouvelle liste de

processus actif

tant que *True* **faire**

- Attendre(Δt unité de temps)
- $B \leftarrow$ Choisirnoeud ()
1. - prépare (S_a)
- envoie S_a à B
- $S_b \leftarrow$ reçoit (B) ;
- $S_a \leftarrow$ MettreAJour (S_a, S_b) ;

fin

processus passif

tant que *True* **faire**

- $S_b \leftarrow$ reçoit $(*)$;
2. - prépare(S_a) ;
- envoie S_a à l'expéditeur de (S_b) ;
- $S_a \leftarrow$ MettreAJour (S_a, S_b) ;

fin

Algorithme 9: Protocole de *gossip*

nœuds, ce qui améliore la diffusion de l'information.

En pratique, chaque nœud maintient une vue partielle du système, c'est-à-dire une table de nœuds nommée voisinage de nœuds. Chaque entrée dans la table contient deux champs principaux : une adresse réseau (adresse *IP* par exemple) d'un nœud et son âge. L'âge peut être exprimé comme le nombre de cycles passés par un nœud depuis qu'il a crée son entrée dans la table d'un autre nœud. Le même nœud peut donc avoir plusieurs âges différents dans les tables des autres nœuds. L'objectif principal de *PSS* est de rafraîchir périodiquement ces tables en utilisant la procédure de *gossip*. Un exemple d'application est illustré dans la figure (4.1).

Deux types d'implémentation de *PSS* ont été proposés dans la littérature : aléatoire et sémantique. Bien que les deux types ont le même but de construire et maintenir la topologie du réseau, le type sémantique consiste à réaliser des topologies spécifiques en regroupant les nœuds similaires selon une fonction sémantique donnée. La plupart des PSS sémantiques utilisent des implémentations aléatoires comme une phase intermédiaire afin de réaliser leurs objectifs. Dans toutes les implémentations, les nœuds appliquent le même protocole de *gossip* illustré en algorithme (9), dans laquelle l'information échangée dans ce cas est une partie de leurs tables. Cependant, chacun d'eux exécute les fonctions de l'algorithme d'une manière différente. En particulier, la façon de choisir les nœuds par la fonction $Choisirnoeud$ et aussi la technique de mettre à jour les tables (fonction $MettreAJour$). Dans la suite, nous allons présenter les différentes techniques appliquées pour ces deux étapes :

– sélection de nœuds : chaque nœud dans les différentes implémentations commence par sélectionner dans sa table un autre nœud, afin d'échanger entre eux une partie de leurs tables de nœuds. Il existe plusieurs façons de choisir ce nœud :

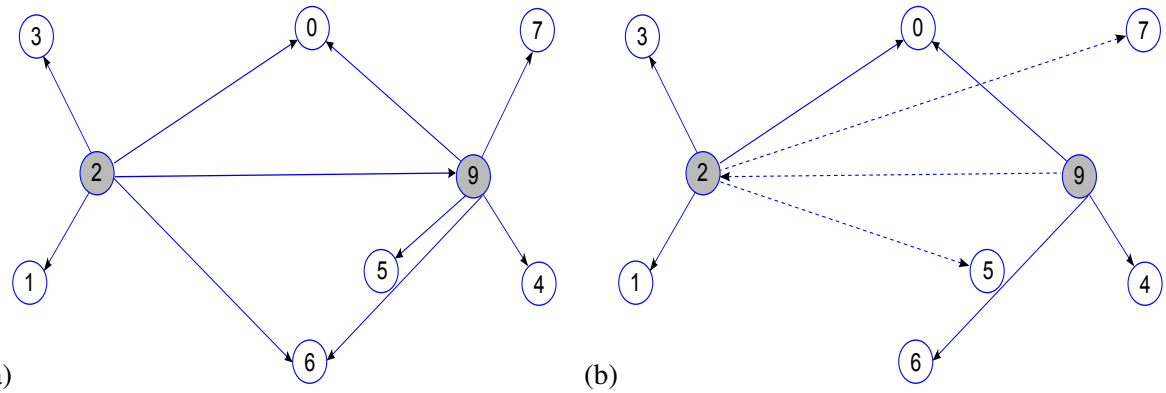


Figure 4.1 – Cette figure est extraite de [116]. Elle montre bien le changement de topologie après l'application du protocole de *cyclon* (une implémentation de *PSS*) sur le nœud 2. Le nœud 2 après avoir choisi aléatoirement son voisin 9, il lui envoie la liste $\{2, 0, 6\}$ et il reçoit également un autre ensemble $\{0, 5, 7\}$. Puis les deux nœuds mettent à jour leurs listes de voisins.

  - soit en sélectionnant aléatoirement un autre nœud disponible à partir de la table de nœud [117],
  - soit en choisissant le nœud qui a le plus grand âge. La raison de contacter ce nœud est d'offrir plus de possibilités pour mettre à jour la table avec des nœuds non connus [116, 68],
  - soit avec le nœud le plus proche sémantiquement dont le but est de former autour de chaque nœud un groupe de nœuds similaires [66, 118] ;
- mise à jour de table : chaque nœud, après la réception de la liste des nœuds d'un autre nœud, met à jour sa table à partir de l'union de nœuds dans sa table et les nœuds reçus. Certains implémentations fixent la taille des tables [117, 68], tandis que les autres n'imposent pas cette contrainte [116], les nœuds dans ces implémentations peuvent avoir des tables de tailles différentes. Dans la plupart des implémentations, chaque nœud élimine, d'abord les doublons en gardant celle avec le plus petit âge. Pour réduire la taille des tables, on peut conserver d'une part, les nœuds de petit âge (récents) [117, 68, 116] et d'autre part, les nœuds les plus proches sémantiquement dans le cas des implémentations sémantiques [66, 118]. Le fait de garder seulement les nœuds les moins âgés permet de donner la priorité aux nœuds qui contiennent les nouvelles informations de rester dans le réseau. Les anciens nœuds sont éliminés progressivement du réseau. Cela permet au réseau de devenir plus robuste contre les défaillances.

Le PSS est caractérisé par plusieurs avantages :

- il modifie dynamiquement la topologie du réseau. Il le rend donc plus dynamique et plus réaliste que les topologies statiques. Ces caractéristiques sont très utiles par exemple dans le cas du réseau *P2P*, dans lequel les nœuds peuvent quitter ou rejoindre le système d'une façon continue, ainsi que pour les défaillances (nœuds, liens) ;
- il sert à propager rapidement l'information dans le réseau d'une façon asynchrone et décentralisée. Il est considéré donc comme important pour les applications distribuées (par exemple agrégation).

Celle-ci est exprimée par le changement continue de topologie du réseau, qui fournit périodiquement pour chaque nœud une nouvelle liste des contacts.

Nous avons présenté dans cette section le service PSS. Il est utilisé dans notre approche proposée (voir le chapitre suivant) pour changer la topologie du réseau et accélérer la convergence.

4.8 Conclusion

Dans ce chapitre, nous avons étudié le problème du clustering dans les environnements distribués. Nous avons présenté pour cela un état de l'art sur les techniques étudiées dans la littérature, en regroupant en même temps les méthodes selon plusieurs aspects différents. Nous avons abordé plus tard dans ce chapitre, le problème de l'estimation distribuée en présence des données atypiques. Pour cette raison, nous avons décrit quelques méthodes statistiques pour réaliser des estimations robustes, ainsi que des techniques de détection des données atypiques. Enfin, nous avons présenté le protocole de *gossip*, utilisé comme un protocole de communication dans notre méthode de clustering.

Ce chapitre forme donc une connaissance basique sur les méthodes de clustering distribuées, en particulier pour introduire notre technique de clustering qui va être proposée dans le chapitre suivant. Elle consiste à estimer un modèle global représentant toutes les données distribuées. Ce modèle est obtenu en fusionnant dynamiquement les modèles locaux (non pas sur les données) générés localement sur les différentes sites. Ainsi, l'estimation du modèle global est réalisée d'une manière robuste, c'est-à-dire en présence des données non fiables. Pour changer la topologie du réseau et accélérer la convergence de l'algorithme, nous avons utilisé une implémentation particulière de *gossip* : le *Peer Sampling Service*.

CHAPITRE 5

Estimation robuste de modèles de mélange sur des données distribuées

5.1 Introduction

La contribution principale de ce chapitre, est une technique d'estimation robuste, proposée pour estimer un modèle de mélange à partir de données distribuées. En particulier, elle est appliquée sur les modèles de mélange gaussien. Elle consiste à fusionner dynamiquement des modèles générés sur des données supposées être réparties sur plusieurs sites différents. Cette méthode prend aussi en compte l'existence des modèles non fiables. Elle réalise donc l'estimation d'une façon robuste, en éliminant ce type de modèles afin d'améliorer la qualité des résultats.

Cette méthode est envisagée pour répondre aux problèmes particuliers étudiés dans le chapitre (1). Rappelons que, avec le développement rapide de l'Internet, ainsi que l'émergence de plusieurs architectures matérielles et logicielles comme des réseaux de capteurs, des réseaux pair-à-pair, il est devenu de plus en plus difficile de réaliser un clustering d'une façon centralisée sur les données distribuées. Cela peut être pour des raisons de respect d'anonymat [80], ou à cause des coûts de transmission et de calcul qui peuvent être très élevés, surtout si les données distribuées sont volumineuses. Il est ainsi plus pratique de communiquer des représentations synthétiques (prototypes des groupes, paramètres de mélanges probabilistes) plutôt que les données. La taille de ces représentations ne dépend pas de celle des échantillons traités.

Notre approche est ainsi basée sur une technique d'agrégation des modèles de mélange. Ce type de méthodes a été très étudié dans les dernières années [114, 46, 42]. Elle réalise un compromis entre la capacité de modéliser une large variété de données et la parcimonie des représentations (nombre limité de paramètres).

Dans notre méthode, l'agrégation des modèles est effectuée seulement sur les paramètres des modèles. Elle est réalisée ainsi d'une façon robuste, c'est-à-dire en présence de modèles non fiables. Pour cela, une technique pour détecter ce type de modèles a été appliquée sur chaque site afin d'obtenir une

agrégation robuste.

Ce chapitre est organisé comme suit : tout d'abord, nous décrivons d'une manière générale notre algorithme de clustering. Ensuite, nous expliquons en détail les différentes étapes de notre algorithme. Nous montrons le critère utilisé pour mesurer la similarité entre les modèles de mélange. Puis les deux techniques proposées pour la détection des modèles non fiables et l'agrégation des modèles sont détaillées. Finalement, le protocole *gossip* utilisé pour changer la topologie du réseau, ainsi que le critère de convergence sont décrits.

5.2 Estimation décentralisée des modèles de mélange

Dans cette section, nous allons décrire tout d'abord l'aspect général de notre approche. Ensuite nous montrons en résumé les différentes étapes de cette approche afin de les détailler dans les sections suivantes.

5.2.1 Aspect général

Notre proposition est un algorithme robuste décentralisé, appliqué indépendamment sur chaque nœud de réseau. Considérons l'existence d'un réseau pair-à-pair composé de n nœuds, dans lequel chaque nœud i est défini avec les propriétés suivantes :

1. un ensemble de données locales D_i ;
2. un modèle de mélange M_i ;
3. une liste de c nœuds voisins L_i ;

Les modèles M_i sont une condition préalable nécessaire à la réalisation de notre algorithme. Ce sont des modèles de mélange probabilistes pouvant être obtenus par un algorithme d'estimation probabiliste, par exemple *EM* [28, 15] ou *EM* variationnelle [15]. Pour la suite nous supposons que chaque ensemble de données D_i est modélisé par un modèle M_i localisé sur le nœud i. La figure (5.1) illustre un exemple d'application de notre technique.

Rappelons que l'objectif principal de notre méthode est de réaliser une estimation robuste du modèle de mélange global, qui reflète la densité de probabilité de toutes les données distribuées. Nous identifions deux genres de robustesse dans notre approche :

– *robustesse en terme d'estimation :* Afin d'améliorer la qualité de résultats et pour réaliser un algorithme robuste en présence de modèles non fiables, il est nécessaire d'identifier et éliminer ces modèles avant d'appliquer la procédure d'agrégation ;

– *robustesse du réseau :* dans le réseau, les liens et les nœuds peuvent tombés en panne (défaillances). Aussi, les nœuds peuvent quitter ou rejoindre le réseau comme dans le cas du réseau pair-à-pair. Ces deux facteurs peuvent influencer fortement les résultats de l'estimation.

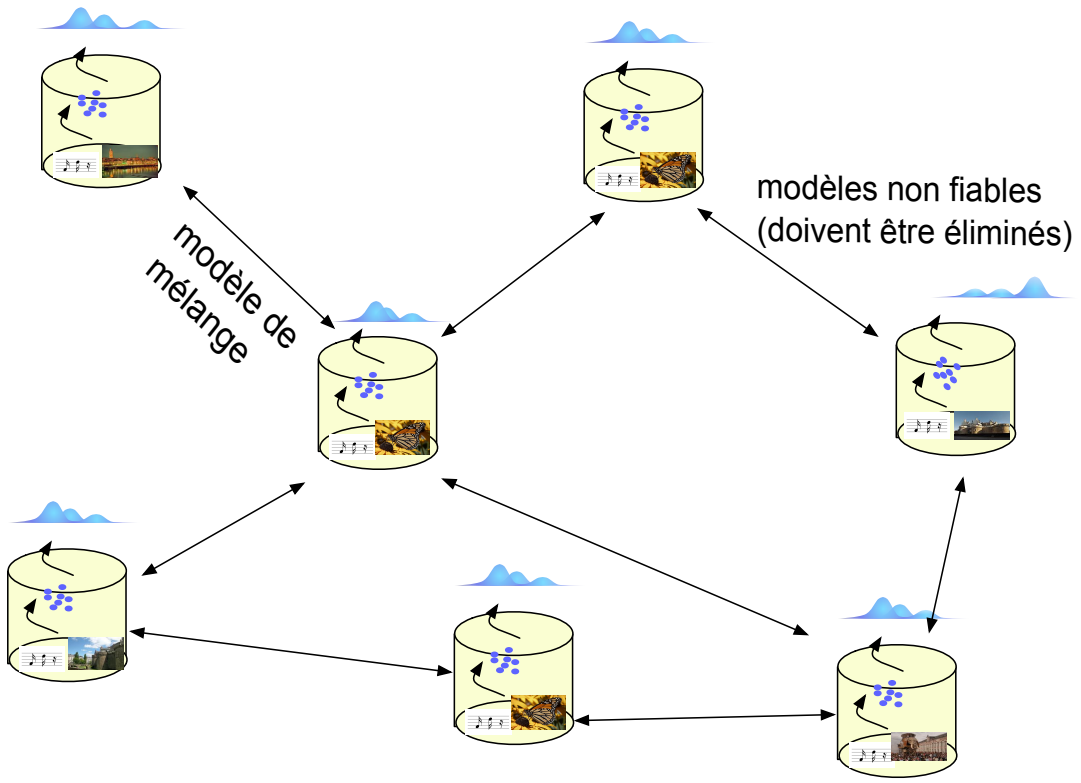


Figure 5.1 – Schéma générale de notre algorithme proposé : d'abord, des modèles de mélange sont estimés localement sur chaque site. Puis, ces modèles sont échangés et agrégés avec les modèles voisins, en utilisant un protocole de communication *gossip*. Notre méthode inclut une étape de détection et d'élimination des modèles non fiables dans le processus d'agrégation.

Notre approche peut gérer ces deux problèmes. D'abord nous avons proposé une technique pour détecter les modèles non fiables. Elle est appliquée pour éliminer ces modèles avant d'effectuer la phase d'agrégation. Nous avons utilisé aussi le protocole de *gossip* qui réalise une communication probabiliste [34, 75], pour faire face au problème de défaillance et de dynamicité du réseau.

5.2.2 Différentes étapes de notre approche

Rappelons que le but de notre algorithme est d'estimer le modèle global représentant les données entières dans le réseau, en fusionnant dynamiquement les modèles locaux M_i. Pour cela, chaque nœud i itère entre les 5 étapes suivantes :

– **récupération des modèles de** L_i : chaque nœud i commence par collecter tous les modèles M_j de ses voisins L_i. Pour la suite, considérons que M_{voisin} est l'ensemble de modèles $L_i \cup M_i$.

Entrées : le modèle M_i, une liste de voisins L_i, le nombre de composantes P du modèle global, et $percent$ le pourcentage de modèles fiables dans le réseau

tant que *(1)* **faire**
  - construire l'ensemble M_{voisin} composé par le modèle M_i et l'ensemble de modèles de ses voisins L_i
  si *la convergence n'est pas réalisée* **alors**
    - éliminer les modèles non fiables dans M_{voisin} afin d'obtenir $M_{robuste}$
    - agréger les modèles de $M_{robuste}$ pour réaliser le nouveau modèle $M_i^{'}$
  fin
  - mettre à jour la liste de voisins avec le *Peer Sampling Service*
fin

Algorithme 10: Processus itératif appliqué sur le nœud i

– **détection des modèles non fiables** : le but de cette étape est d'éliminer les modèles non fiables existant dans l'ensemble M_{voisin} afin de construire un ensemble robuste de modèles noté $M_{robuste}$. L'algorithme appliqué ici est une adaptation de l'algorithme proposé dans [100]. Il se base aussi sur l'échantillonnage pour réaliser son objectif, mais les objets manipulés dans ce cas sont des modèles et non pas des données. Notons que le modèle local M_i du nœud i peut être aussi considéré comme un modèle non fiable.

– **mise à jour du modèle local** : après la calcul de $M_{robuste}$, le nœud i applique un algorithme d'agrégation sur l'ensemble de modèles de $M_{robuste}$ afin de construire son nouveau modèle. Pour effectuer l'agrégation, nous utilisons la même technique proposée dans [47]. Elle fonctionne de la même façon que l'algorithme *K-means :* étant donné l'ensemble des modèles à agréger $M_{robuste}$, et le nombre de composantes du nouveau modèle P, elle consiste à regrouper les composantes similaires de $M_{robuste}$ en minimisant la divergence entre les modèles de $M_{robuste}$ et le modèle réduit. Le modèle réduit obtenu forme le nouveau modèle du nœud i.

– **application du protocole de communication *gossip*** : le protocole *gossip* est par définition une technique pour diffuser aléatoirement l'information (modèle dans notre cas) dans un réseau. Il peut donc réaliser une propagation correcte du modèle M_i due à sa façon probabiliste de communiquer l'information. Nous utilisons une implémentation de ce protocole : le *Peer Sampling Service*. Il consiste à changer périodiquement les voisins de chaque nœud, en d'autres termes la topologie du réseau. Ainsi, il accélère la convergence des modèles locaux vers le modèle global. Chaque nouvelle agrégation est effectuée sur un ensemble différent de modèles qui correspondent aux nouvelles listes de voisins.

– **vérification du critère de convergence** : avant de commencer une nouvelle itération, le nœud i compare son modèle avec les modèles de ses voisins. Selon les résultats obtenus, il exécute soit une nouvelle itération de l'algorithme, soit une application de PSS.

L'algorithme général et un exemple de différentes étapes sont présentés respectivement dans l'algorithme (10) et la figure (5.2). Cette figure montre bien la succession d'étapes sur le nœud (1). Après avoir estimer son modèle gaussien, le nœud (1) récupère les modèles de ses voisins. Puis, il applique une

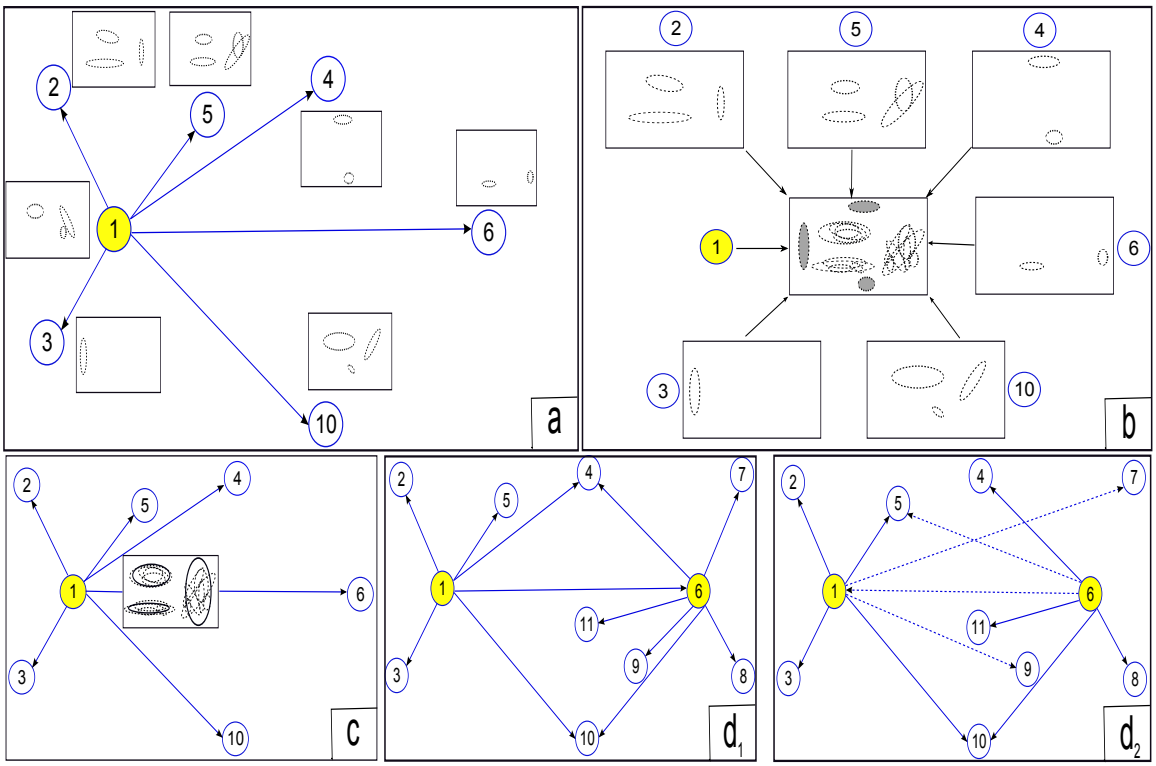


Figure 5.2 – Exemple de notre approche robuste et décentralisée sur une partie d'un graphe du réseau : la figure (a) montre les modèles initiaux du nœud (1) et ses voisins, composé chacun par un nombre différent des composantes gaussiennes. Puis tous les modèles gaussiens de L_1 sont collectés par le nœud (1), ils sont représentés par les ellipses en pointillé (figure (b)). Une technique pour la détection des modèles non fiables est effectuée sur cet ensemble de modèles : le résultat de cette détection est illustré dans la figure (b) par les modèles composés des ellipses pleines en pointillé. Ces modèles des nœuds 3 et 4 sont considérés comme non fiables. Ils doivent être éliminés avant l'application de la phase d'agrégation. Les ellipses en noir dans la figure (c) représentent le nouveau modèle du nœud (1) obtenu en agrégeant les modèles restants. Enfin, le nœud (1) échange une partie de ses voisins avec le nœud (6) (figures d_1 et d_2), en appliquant la technique de PSS.

technique pour éliminer les modèles non fiables. Ensuite, un algorithme d'agrégation est appliqué sur les modèles restants. Le but est de générer le nouveau modèle du nœud (1). Enfin, le nœud (1) choisit un autre nœud parmi la liste de ses voisins, afin d'échanger entre eux une partie de leurs voisins. Ces étapes correspondent à une itération de l'algorithme (10) sur le nœud (1). Après la vérification du critère de convergence, le nœud (1) applique seulement le PSS ou ré-exécute une itération de l'algorithme.

Après avoir expliqué les étapes principales de notre algorithme, nous montrerons dans la section suivante le critère utilisé pour mesurer la similarité entre les modèles de mélange.

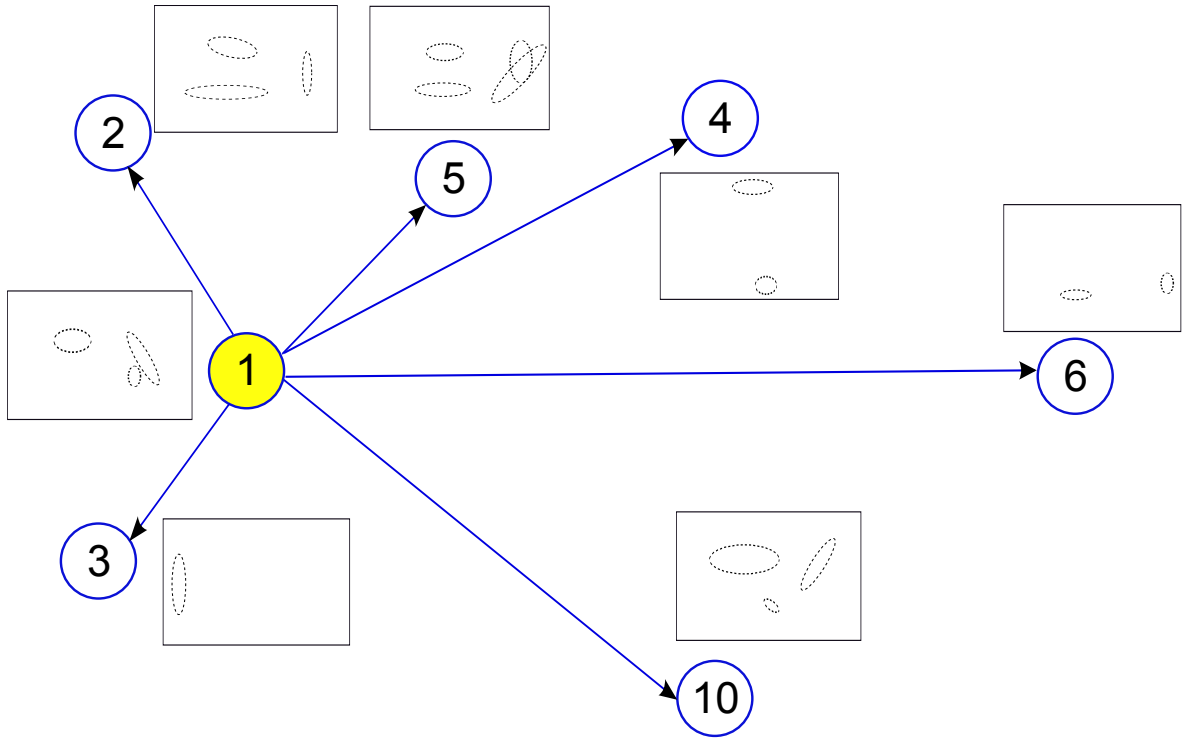


Figure 5.3 – Cette figure présente une partie du graphe d'un réseau coupé au niveau du nœud (1). Les ellipses en pointillé sur chaque nœud représentent les modèles correspondants. Le nœud (1) possède ici une liste de 6 voisins. Nous allons voir après, comment est obtenu le modèle global sur cet exemple.

5.2.3 Similarité entre les modèles de mélange gaussien

Dans notre algorithme, nous avons besoin de calculer la similarité entre les mélanges gaussiens tout au long de l'exécution des étapes 2 et 3 de notre algorithme (10). En effet, les mélanges gaussiens doivent être comparés entre eux afin de :

- identifier les modèles non fiables (étape 2 du l'algorithme (10)). Pour cela, chaque nœud collecte dans un premier temps les modèles de ses voisins. Puis il cherche à trouver parmi eux ceux qui s'écartent le plus par rapport à la majorité des modèles. Cela nécessite de calculer la similarité entre chaque paire de modèles collectés ;

- réaliser la phase d'agrégation (étape 3 du l'algorithme (10)). Cette phase a pour but de réduire un modèle de mélange en un autre modèle avec un nombre plus petit de composantes. Elle nécessite aussi le calcul de similarité entre les modèles originaux et le modèle réduit.

Nous avons proposé la divergence de *Kullback-Leibler* (mesure fondamentale entre les modèles de mélange), comme un critère de similarité entre les mélanges gaussiens. Ce critère qui a été présenté en détail dans le chapitre 3 (section 3.3) va être rappelé dans ce chapitre. Nous avons vu que la divergence de KL n'existe pas sous forme déterminée entre deux mélanges gaussiens, et que plusieurs approximations ont été proposées pour calculer cette divergence. Notre choix s'est porté sur la méthode de *match bound* [45] pour calculer l'approximation de *KL* entre les mélanges gaussiens. L'utilisation de cette approximation était motivée par sa complexité de calcul réduite (seuls les paramètres des modèles sont requis). Nous proposons ainsi de réutiliser ce critère afin de calculer la similarité entre les mélanges gaussiens.

En résumé, la méthode de *match bound* consiste à calculer la divergence entre deux mélanges f et g comme suit :

$$KL_{match}(f||g) = \sum_i \pi_{f_i} \left(KL(f_i||g_{m(i)}) + log \frac{\pi_{f_i}}{\pi_{g_{m(i)}}} \right), \tag{5.1}$$

où π_{f_i} et $\pi_{g_{m(i)}}$ sont respectivement les probabilités a priori des composantes f_i et $g_{m(i)}$. m est la fonction de correspondance entre les différentes composantes de f et g définie par l'équation (3.15). KL est la divergence entre deux composantes gaussiennes calculée selon l'équation (3.13).

Pour résumer, ce critère est utilisé comme une mesure de similarité entre les mélanges gaussiens afin de réaliser la détection des données non fiables et l'agrégation des modèles. Dans la section suivante, nous montrerons comment sont détectés les modèles non fiables.

5.3 Détection des modèles non fiables

Les modèles non fiables peuvent être produits soit à partir de données atypiques, soit à partir d'une estimation fausse. Ils peuvent affecter fortement les résultats de l'estimation. Il faut donc les éliminer avant de réaliser la phase d'agrégation sur les modèles.

Entrées : Un ensemble de mélanges gaussiens $M_{voisin} = \{M_1, ..., M_L\}$ et $percent$ le pourcentage de modèles fiables dans le réseau

- Choisir aléatoirement T échantillons $\{S_1, ..., S_T\}$, composé chacun par $S_{taille} = (percent * L)$ modèles de M_{voisin}
- **pour** *Chaque échantillon* S_t, $1 \leq t \leq T$, *répéter les opérations suivantes deux fois* **faire**
  - **pour** *Chaque modèle* $M_j \in M_{voisin}$, $1 \leq j \leq L$ **faire**
    - Calculer la divergence KL_{match} entre M_j et M_{agreg} le modèle de concaténation de tous les modèles $M_r \in S_t$, $M^{agreg} = \frac{1}{\pi} \sum_{r=1}^{S_{taille}} M_r$, où $M_r = \sum_{i=1}^{m_i} \pi_r^i N^i$, $\pi = \sum_{r=1}^{S_{taille}} \pi_r^i$, et N^i est une composante gaussienne
  - **fin**
  - $S_t = \{ M_r \mid r = \arg\min_{1 \leq j \leq L} (KL_{match}(M^{agreg}, M_j))\}$, où $|S_t| = S_{taille}$
- **fin**
- Sélectionner parmi $\{S_1, ..., S_T\}$, les T' meilleurs échantillons $S_{meilleur} = \{S_1, ..., S_{T'}\}$ qui minimisent $\sum_{j=1}^{L} KL_{match}(M^{agreg}, M_j)$
- **pour** *Chaque échantillon* $S_t \in S_{meilleur}$, $1 \leq t \leq T'$ **faire**
  - **répéter**
    - **pour** *Chaque modèle* M_j, $1 \leq j \leq L$ **faire**
      - Calculer $KL_{match}(M^{agreg}, M_j)$
    - **fin**
    - $S_t = \{ M_r \mid r = \arg\min_{1 \leq j \leq L} (KL_{match}(M^{agreg}, M_j))\}$ et $|S_t| = S_{taille}$, S_t est remplacé par les modèles qui ont les minimales divergences avec M_{agreg}
  - **jusqu'à** S_t *ne change pas*;
- **fin**

Sorties : S_t le meilleur échantillon dans $S_{meilleur}$ qui minimise $(\sum_{j=1}^{L} KL_{match}(M^{agreg}, M_j))$. Cet échantillon contient les S_{taille} modèles les plus proches entre eux

Algorithme 11: Algorithme d'échantillonnage pour exclure les modèles non fiables (mal estimés ou représentant des données atypiques) dans un ensemble des modèles de mélange.

Notre proposition est une adaptation de l'algorithme d'échantillonnage DCM (Déterminant de Covariance Minimal) proposé dans [100] pour la détection des données atypiques. L'objectif de cet algorithme est de trouver parmi l'ensemble des données, un échantillon qui ne contient pas de données atypiques (isolées), puis d'estimer le modèle mathématique à partir de cet échantillon. Le modèle calculé est considéré comme robuste, parce qu'il est obtenu seulement à partir des données supposées normales.

Tout d'abord, nous commençons par décrire les étapes principales de la technique proposée dans [100], puis nous allons expliquer dans un deuxième temps comment adapter cette technique dans notre contexte, ainsi que les avantages et les inconvénients.

5.3.1 Déterminant de covariance minimal (DCM)

La méthode (DCM) est une technique d'estimation robuste [100, 79], développée pour calculer le modèle statistique (moyenne, variance) d'un ensemble de données, en présence des données atypiques. Elle peut être utilisée dans plusieurs applications, nécessitant des estimations robustes. Par exemple, pour réaliser un clustering robuste et identifier en même temps les données atypiques [57], ou pour effectuer

de classification robuste en remplaçant la matrice de covariance classique par celle de (DCM) [101]. Dans ce chapitre, nous adaptons cette méthode pour qu'elle fonctionne sur les modèles.

Par définition, la technique de (DCM), consiste à estimer la moyenne et la matrice de covariance à partir d'un échantillon de points, qui minimise le déterminant de la matrice de covariance correspondante [100, 79]. Autrement dit, il commence par sélectionner parmi l'ensemble des données étudiées, plusieurs échantillons de points. Puis un algorithme itératif est appliqué sur ces échantillons afin de les améliorer, et de choisir ceux contenant les données les plus représentatives (les plus proches entre elles). La moyenne et la matrice de covariance de cet échantillon sont les estimateurs (DCM) de la localisation et de la dispersion des données. Dans la suite, nous allons décrire l'algorithme (FAST-DCM) développé pour réduire la complexité de calcul et accélérer la convergence de l'algorithme standard (DCM) [100, 79] :

- il commence par sélectionner un nombre important d'échantillons de données de h points chacun, à partir d'un ensemble de n points $\{x_i\}_{i=1}^{n}$. h est un paramètre utilisateur, sa valeur par défaut est $h = (n+p+1)/2)$, où p est la dimension des données. En pratique, il est initialisé à $0.75n$ [100] ;

- pour chaque échantillon H_i de moyenne μ_i et de matrice de covariance Σ_i, répéter les étapes suivantes deux fois :

  1. calculer la distance de *Mahalanobis* entre chaque point x_i et μ_i [100],

  2. remplacer H_i par les h points qui ont les distances les plus proches selon l'étape précédente,

  3. re-calculer la moyenne et la covariance de H_i à partir de nouveaux points associés ;

- choisir les 10 meilleurs échantillons qui ont le déterminant de matrices de covariances minimal [100], puis relancer la procédure sur ces échantillons jusqu'à la convergence. La convergence est réalisée lorsque le déterminant de la matrice de covariance entre deux changements consécutifs de H_i reste stable.

Nous avons décrit dans cette section, le principe de l'algorithme (DCM). Dans la suite, nous allons détailler notre proposition pour la détection des modèles non fiables en se basant sur cette approche.

5.3.2 Adaptation de (DCM) sur les modèles de mélange

Après avoir expliqué la technique de (DCM) dans la section précédente, nous allons montrer comment adapter cette technique sur les modèles de mélange. D'abord, notons que cette technique est appliquée localement sur chaque nœud, sur l'ensemble des modèles collectés à partir de ses voisins. Les différences par rapport à la technique de (DCM) standard sont les suivantes :

- les objets manipulés dans notre méthode sont les modèles locaux générés sur les différents nœuds et non pas les données. Le but ici est donc de trouver parmi l'ensemble des modèles collectés, l'échantillon qui représente le mieux ces modèles, c'est-à-dire celui qui contient les modèles les

plus proches entre eux. Par conséquent les modèles non fiables sont éliminés à cause de leurs distributions qui s'écartent significativement de la majorité des modèles ;

– la similarité entre les modèles est mesurée par la divergence de KL_{match} (équation (3.16)), et non pas les distances métriques (distance de *Mahalanobis* comme dans [100]).

L'algorithme (11) décrit les différentes étapes de notre technique pour la détection des modèles non fiables. D'abord, chaque nœud collecte tous les modèles de ses voisins. Puis, il commence par choisir aléatoirement T échantillons de taille S_{taille} à partir de l'ensemble de modèles collectés $M_{voisin} = \{M_1, \ldots, M_j, \ldots, M_L\}$. Ensuite, pour chaque échantillon $S_t = \{M_1, \ldots, M_{S_{taille}}\}$, $t \in [1, T]$, les étapes suivantes sont appliquées deux fois (étape 3 à 6) :

1. fusionner tous les modèles de l'échantillon S_t, pour former le modèle M_{agreg}. En d'autres termes, M_{agreg} est le modèle de concaténation de tous les modèles M_r dans S_t, mais en normalisant leurs poids ;

2. calculer la divergence KL_{match} entre M_{agreg} et chaque modèle dans M_{voisin}. Puis les modèles sont ordonnés d'une façon croissante selon les valeurs de divergence obtenues (étape 3 à 5 de l'algorithme 11) ;

3. remplacer les modèles M_j de l'échantillon S_t, par les premiers S_{taille} modèles de la liste obtenue avant (étape 6).

L'algorithme a été appliqué d'abord deux fois sur chaque échantillon. Le but est de réduire la complexité de calcul, due au nombre d'échantillons générés qui peut être important. Ensuite, l'algorithme est complété, en sélectionnant d'abord un ensemble très petit des échantillons $S_{meilleur} = \{S_t\}, 1 \leq t \leq T'$ (étape 8). La procédure (étape 3 à 6) est relancée sur chaque S_t dans $S_{meilleur}$ jusqu'à la convergence (étape 9 à 16). La convergence est réalisée lorsque chaque échantillon S_t dévient stable. Les échantillons dans $S_{meilleur}$ sont ceux qui minimisent la divergence KL_{match} avec les modèles M_j de M_{voisin} (étape 8). Finalement, l'échantillon S_t de $S_{meilleur}$, qui réalise le minimum de divergence avec les modèles de M_{voisin} est considéré comme le meilleur échantillon parmi l'ensemble des échantillons traités. Les modèles de cet échantillon sont utilisés ensuite comme une entrée pour la procédure d'agrégation. La figure (5.4) illustre le déroulement de l'algorithme (11) pour la détection de modèles non fiables.

Cette technique est caractérisée et motivée par plusieurs propriétés :

– elle ne dépend pas de l'effet échelle des espaces de données. Celle-ci est considérée comme un problème majeur pour certaines méthodes comme les méthodes basées sur la distance entre les données, dans laquelle il est difficile de fixer un seuil pour la détection des données atypiques ;

– elle est effectuée d'une façon décentralisée et locale. Elle est donc caractérisée par une complexité de calcul réduite en comparaison avec la méthode (DCM) sur les données [100]. Celle-ci est exprimée par le fait que le nombre de modèles collectés sur chaque nœud est très petit par rapport au nombre de données étudiés dans [100]. La figure (5.5) montre par exemple l'application décentra-

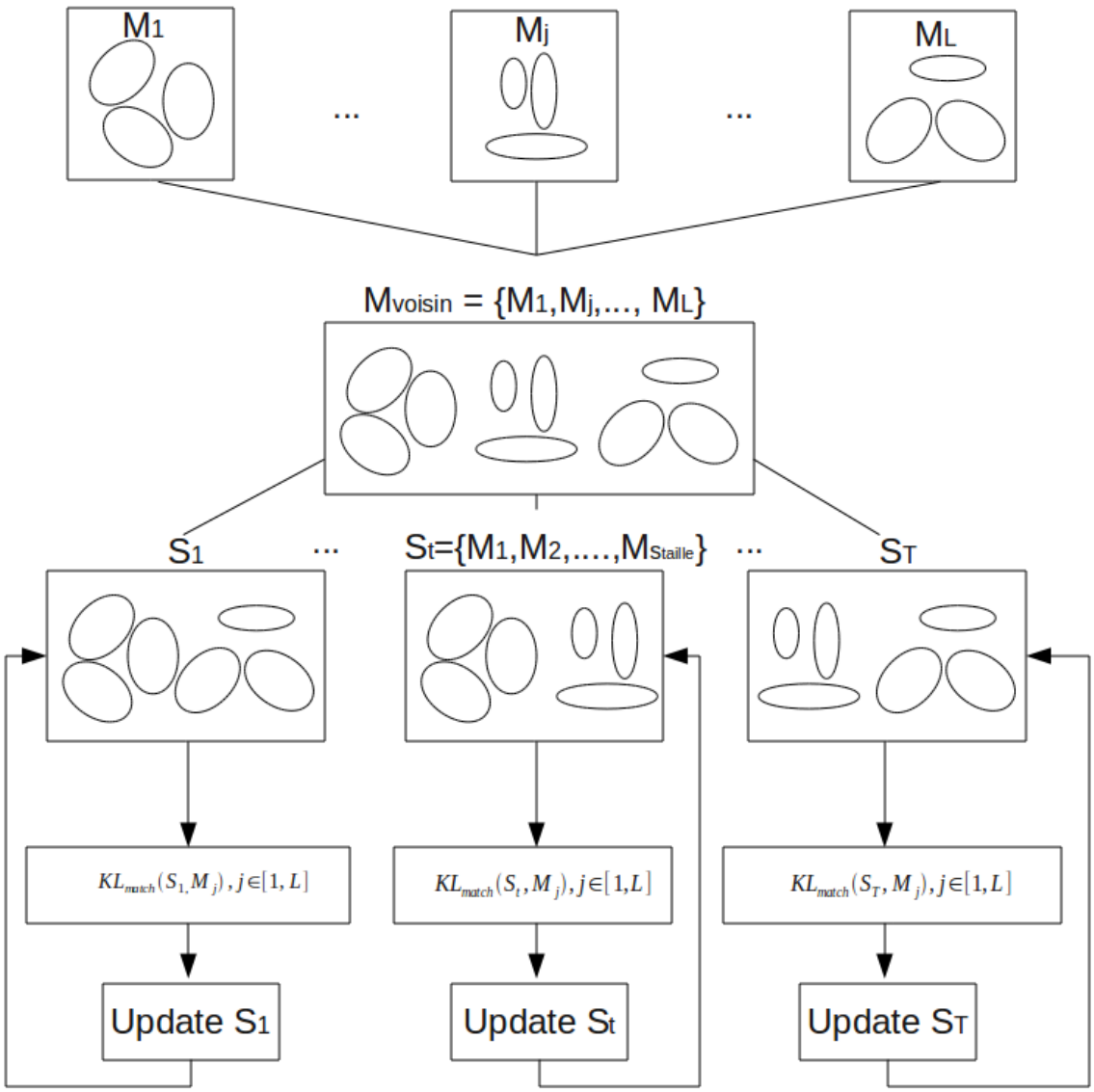


Figure 5.4 – Exemple de l'algorithme d'échantillonnage pour la détection de données non fiables sur le nœud i : d'abord, le nœud i commence par collecter tous les modèles de ses voisins. Ensuite, il génère aléatoirement T échantillons composés chacun de S_{taille} modèles à partir de l'ensemble $M_{voisin} = \{M_1, \ldots, M_L\}$. Pour chaque échantillon, un algorithme itératif est appliqué pour améliorer sa qualité, en éliminant les modèles non fiables. Cet algorithme consiste à comparer la similarité entre les modèles $M_j \in M_{voisin}$ et ceux dans S_t. Puis les modèles M_j les plus proches sont sélectionnés pour former à nouveau l'échantillon S_t. La convergence est réalisée lorsque tous les échantillons deviennent stables

lisée de notre technique sur le nœud (1).

Dans la section suivante, nous montrerons quels sont les paramètres qui peuvent influencer les résultats de l'algorithme.

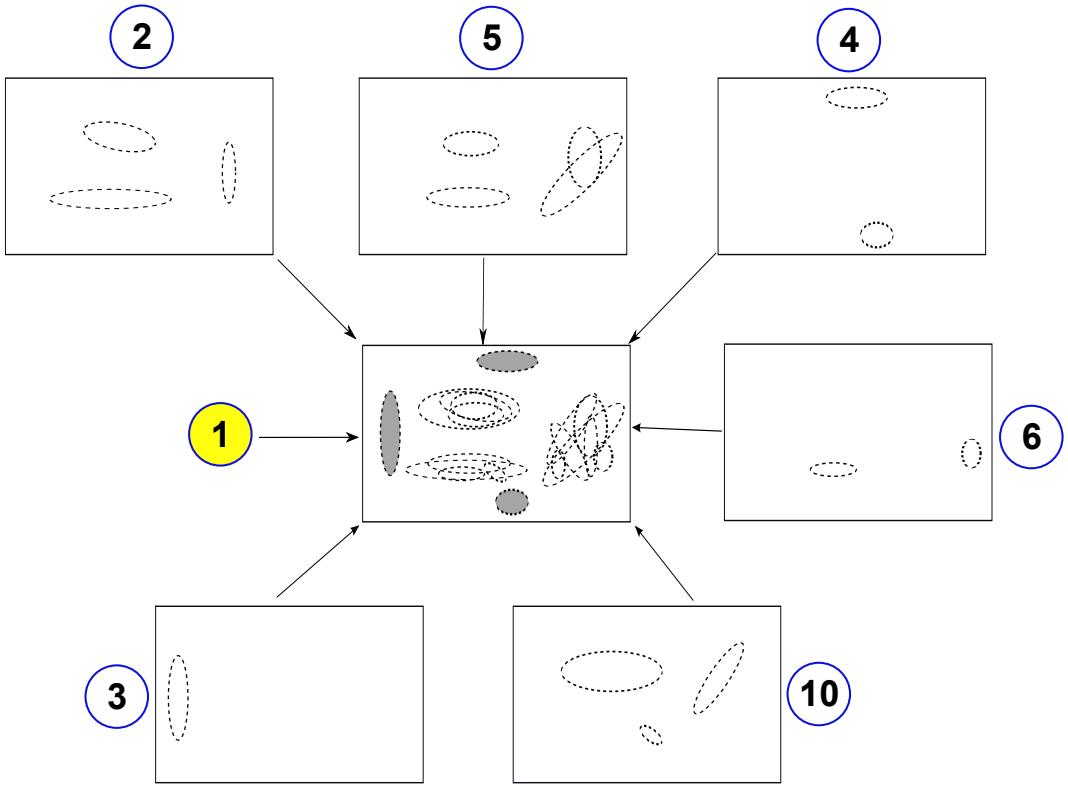


Figure 5.5 – Cette figure présente un exemple de détection de modèles non fiables, appliqué sur notre exemple précédent (figure 5.3). Le nœud (1), après avoir collecté les modèles de ses voisins, élimine les modèles non fiables (modèles des nœuds 3 et 4 dans cet exemple). Les modèles non fiables sont ceux qui s'écartent de la majorité des autres modèles, ils sont représentés par les ellipses en gris. Les modèles restants sont utilisés par l'étape d'agrégation.

5.3.3 Impact des paramètres

Cet algorithme dépend de plusieurs paramètres : $percent$ le pourcentage des modèles fiables dans le réseau, le nombre des échantillons T générés sur chaque nœud, la taille S_{taille} de chaque échantillon :

– paramètre T : c'est pour préciser la quantité des échantillons à générer sur chaque nœud. Il peut jouer aussi un rôle important sur la qualité des résultats. En pratique, si le nombre T est élevé, alors le nombre des échantillons à tester est grand. Par conséquent la probabilité d'avoir un échantillon robuste, c'est-à-dire ne contenant pas de modèles non fiables, augmente. Cependant une valeur grande de T nécessite plus de calcul. Rappelons qu'une solution réduite pour améliorer ce point a

été proposée dans l'algorithme (11)(étape 2 à 7) ;

– paramètre $T^{'}$: c'est le nombre de meilleurs échantillons sélectionnés parmi l'ensemble des échantillons testés. Rappelons que ces échantillons contiennent des modèles supposés proches entre eux. Le paramètre $T^{'}$ est initialisé avec une petite valeur (10 en pratique). Notons qu'une valeur grande augmente le nombre d'échantillons à tester, ce qui nécessite plus de calculs. En revanche, une valeur très petite peut diminuer la probabilité d'obtenir le meilleur échantillon final, c'est-à-dire l'échantillon qui contient les modèles les plus représentatifs ;

– paramètre $percent$: il sert à estimer a priori le pourcentage de modèles fiables, et par conséquent les modèles non fiables. Il a été utilisé dans cet algorithme pour déterminer localement sur chaque nœud, le nombre de modèles à sélectionner parmi ceux qui sont collectés à partir de ses voisins (taille des échantillons). Par exemple, s'il est initialisé à 75%, alors il faut choisir ce pourcentage de modèles parmi l'ensemble M_{voisin}. Les 25% de modèles restants sont considérés comme des modèles non fiables.

  Si la valeur de $percent$ est grande, alors il y a plus de chance que l'échantillon contienne des modèles non fiables. Inversement, si la valeur est petite, cela peut entraîner à ne pas choisir tous les modèles fiables. La conséquence était une perte d'informations (mauvais modèle global). Donc l'initialisation de ce paramètre est un compromis entre la robustesse et la qualité de l'échantillon final. Notons que même avec une valeur fausse, le problème peut être limité en choisissant un nombre important d'échantillons T. Cela augmente la probabilité d'avoir le meilleur échantillon final.

Après avoir appliqué notre technique d'échantillonnage pour éliminer les modèles non fiables, nous allons montrer dans la section suivante la procédure d'agrégation.

5.4 Agrégation de modèles de mélange

Nous avons vu dans la section précédente que la méthode de détection des modèles non fiables, fournit en sortie un échantillon de modèles supposé ne contenir aucun modèle non fiable. Ces modèles vont être agrégés pour estimer le nouveau modèle d'un nœud donné. Dans la suite, nous expliquons en détail notre technique pour agréger les modèles de mélange.

Le problème d'agrégation d'un modèle de mélange, consiste à transformer un modèle de mélange f en un autre mélange g avec un nombre plus petit de composantes. Le modèle f dans notre cas est construit à partir des modèles de l'échantillon résultant de notre sélection des modèles fiables.

La technique utilisée ici pour la réduction de modèle de mélange est la méthode de *Goldberger et al.* [47]. Cette technique détaillée dans le chapitre (3), est caractérisée par un coût de calcul réduit. Elle fonctionne de la même façon que l'algorithme *K-means*, mais plutôt sur les paramètres de composantes que sur les données. Le seul changement effectué est que l'approximation KL_{match} est utilisée pour calculer la divergence entre les deux mélanges f et g.

Rappelons que le modèle réduit g est obtenu en alternant entre les deux étapes suivantes :

1. la première étape consiste à déterminer la meilleure fonction de correspondance m entre les composantes de f et g, en minimisant le critère suivant :

$$\begin{aligned} d(f,g) &= d(f,g,m) \\ &= KL_{match}(f,g) \\ &= \sum_i \pi_{f_i} \left(KL(f_i||g_{m(i)}) + log\frac{\pi_{f_i}}{\pi_{g_{m(i)}}} \right) \end{aligned} \tag{5.2}$$

   où $m = \arg\min_\phi \quad d(f,g,\phi)$, et $\phi : \{1,\cdots,K\} \rightarrow \{1,\cdots,K^{'}\}$ est la fonction de correspondance entre les composantes de f et g.

2. dans la deuxième étape, les paramètres de g sont mis à jour à partir des paramètres du modèle f seulement. Les moyennes et les variances des composantes de g sont donc calculées comme la moyenne des celles de composantes de f selon la fonction de correspondance m obtenue dans l'étape précédente. Chaque composante gaussienne de g est mise à jour comme suit :

$$\begin{aligned} \mu_{g_j} = & \quad \frac{1}{\pi_{g_j}} \sum_{i \in m^{-1}(j)} \pi_{f_i} \mu_{f_i} \\ \Sigma_{g_j} = & \quad \frac{1}{\pi_{g_j}} \sum_{i \in m^{-1}(j)} \pi_{f_i} \\ & \left((\Sigma_{f_i} + (\mu_{f_i} - \mu_{g_j})(\mu_{f_i} - \mu_{g_j})^T) \right) \end{aligned}$$

   où $\pi_{g_j} = \sum_{i \in m^{-1}(j)} \pi_{f_i}$.

Ces deux étapes sont itérées jusqu'à la convergence du critère défini dans l'équation (5.2). Un exemple d'application est présenté dans la figure (5.6).

Rappelons que notre technique d'agrégation présente les mêmes inconvénients que l'algorithme *K-means* (le nombre de composantes doit être précisé à l'avance, initialisation des paramètres). Le nombre de composantes $K^{'}$ du modèle réduit reste le seul paramètre critique de notre algorithme. Cependant nous utilisons la technique de *K-means++* pour résoudre le problème d'initialisation *K-means++*. Cette méthode *K-means++*, qui a été implémentée à la base pour fonctionner sur les données *K-means++*, est adaptée ici sur les modèles. Cette méthode consiste à choisir les centres initiaux les plus éloignés entre eux. Les centres choisis dans notre algorithme sont les composantes gaussiennes, et la distance utilisée est la divergence KL. Les différentes étapes de l'algorithme *K-means++* appliquées ici, sont les mêmes que celles dans l'algorithme (7) du chapitre (3). La seule différence est que les composantes de la distribution de *Student* sont remplacées par des gaussiennes. Les centres obtenus sont utilisés ensuite comme des paramètres d'initialisation pour l'algorithme d'agrégation.

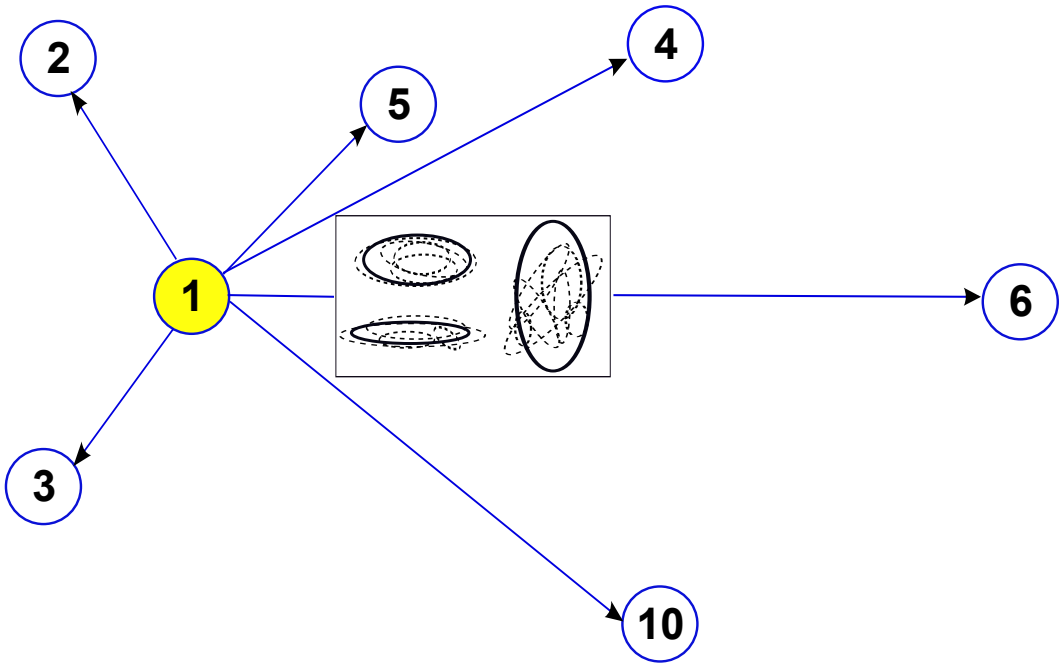


Figure 5.6 – Cette figure montre bien la phase d'agrégation appliquée sur le nœud (1). Toutes les composantes similaires sont regroupées en une seule composante, et le modèle obtenu constitue le nouveau modèle de ce nœud (ellipses en noir).

Une fois que les nouveaux modèles sont estimés sur les différents nœuds, un protocole de communication est requis pour échanger les modèles. Dans la suite, nous allons présenter notre technique pour échanger les modèles.

5.5 Mise à jour de la topologie de réseau

Pour améliorer la diffusion de l'information (modèles dans notre contexte) dans un réseau, nous avons proposé d'utiliser le *Peer Sampling Service* (PSS) [82, 68, 116]. C'est une implémentation particulière de protocole de *gossip* [34, 75]. Il consiste à changer dynamiquement la topologie du réseau en fournissant périodiquement à chaque nœud de nouvelles listes de voisins (voir la section (4.7) du chapitre 4).

Le PSS est caractérisé par plusieurs avantages :

– il assure une diffusion fiable de l'information (données, modèles, ..). Celle-ci est due à sa façon probabiliste d'échanger cette information entre les nœuds. Il ne cause pas donc une dégradation importante sur les résultats, en particulier si les nœuds rejoignent ou quittent le réseau, aussi bien

dans le cas de défaillance des nœuds et de liens ;

– il peut accélérer exponentiellement la convergence de l'algorithme [67, 13]. Cela est exprimé par le changement aléatoire de topologie du réseau, qui réalise périodiquement pour chaque nœud, une nouvelle liste de voisins. Par conséquent, il fournit à ce nœud de nouveaux modèles pour les agréger.

Plusieurs implémentations de *PSS* ont été étudiées dans le chapitre (4). Rappelons qu'il y a deux types d'implémentations : aléatoire et sémantique (section 4.7.2). Ici, nous n'avons pas besoin de regrouper les nœuds d'une façon sémantique, mais le but est de diffuser les modèles agrégés aléatoirement. Nous utilisons donc une implémentation aléatoire en considérant que chaque nœud i maintient une liste de ses c voisins. Chaque nœud était identifié par un numéro (id). Ce protocole exécute les étapes suivantes :

1. il commence par sélectionner aléatoirement un nœud j à partir de sa liste de voisins, afin d'initialiser la communication ;

2. il envoie à j la moitié de sa liste de voisins en y incluant son id, et il reçoit simultanément de j une liste équivalente du même nombre de nœuds (ne contient pas forcément l'id de j). Les nœuds sélectionnés pour l'échange sont choisis aléatoirement parmi les listes de voisins. L'id de i est envoyé afin d'assurer la connectivité dans le réseau (i sera dans la liste de voisins de j) ;

3. il met à jour sa liste de voisins à partir de l'union de sa liste courante et celle reçue de j. Pour cela, i et j réalisent les étapes suivantes. Pour le nœud i, les actions à effectuer sont :

   – il élimine les doublons. En d'autres termes, si un nœud k apparaît deux fois dans la liste des voisins de i, alors il faut supprimer un de ces occurrences,

   – il garde les nouveaux nœuds reçus, c'est-à-dire ceux qui ne sont pas dans la liste initiale des voisins de i. La topologie de réseau est donc modifiée,

   – il complète la liste de nœuds obtenue dans l'étape précédente pour arriver à une taille c. Celle-ci est réalisée en ajoutant d'autres nœuds sélectionnés aléatoirement de la liste initiale.

L'algorithme (12) présente le squelette de notre implémentation *PSS*. Il décrit les étapes effectuées par un nœud i qui joue le rôle d'un nœud actif, et un autre nœud j qui joue le rôle passif. Un nœud dans l'état passif, peut répondre à plusieurs nœuds dans l'état actif, et échanger des nœuds avec eux. Un exemple d'application de *PSS* est illustré dans la figure (5.7), qui montre bien le changement de topologie du réseau après l'échange de nœuds voisins entre deux nœuds.

Le service PSS accélère donc la convergence du modèle global. Il améliore aussi la qualité des résultats, grâce au changement des voisins qui évite la concentration des modèles non fiables. En d'autres termes, les modèles non fiables sont dispersés aléatoirement dans le réseau. Cela augmente la probabilité de les détecter en appliquant notre technique d'échantillonnage.

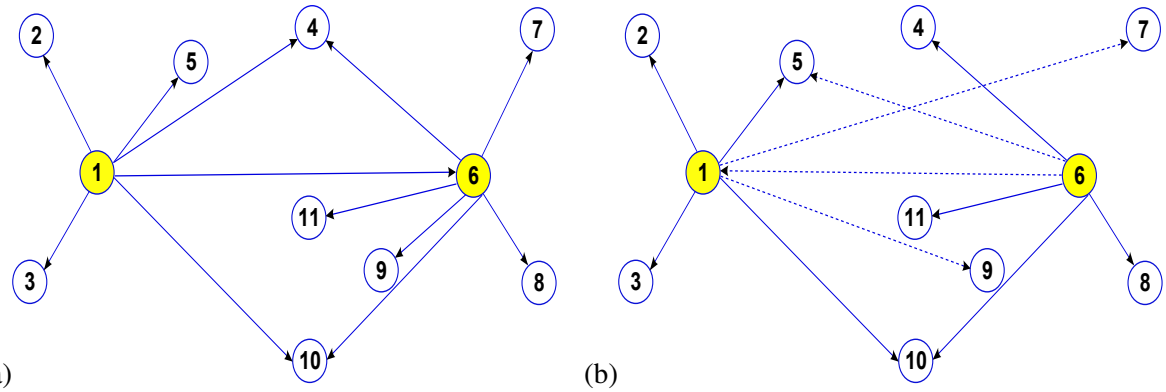
Nœud actif

1. - Choisir aléatoirement un nœud j à partir de sa liste de voisins L_i
2. - Créer une liste $L_i^{'}$, en ajoutant dans un premier temps le nœud i à cette liste, puis de la compléter par $(c/2-1)$ nœuds sélectionnés aussi aléatoirement de L_i
3. - Envoyer cette liste à j
4. - Recevoir de j une liste $L_j^{'}$ de $c/2$ nœuds
5. - Réduire l'union de deux listes $L_i \cup L_j^{'}$ en revenant à c

Nœud passif

1. - Attendre jusqu'à la réception d'une liste $L_i^{'}$ d'un autre nœud
2. - Préparer la liste $L_j^{'}$, en sélectionnant aléatoirement $c/2$ nœuds à partir de L_j
3. - Envoyer $L_j^{'}$ à l'expéditeur de $L_i^{'}$
4. - Réduire $L_j \cup L_i^{'}$ en revenant à c

Algorithme 12: Cet algorithme présente le rôle d'un nœud dans les deux états (actif et passif) de notre implémentation PSS, proposé pour changer la topologie du réseau à la fin de chaque itération de notre l'algorithme.



(a) (b)

Figure 5.7 – Cette figure montre l'application de *PSS* par le nœud (1). Les nœuds voisins de (1) sont représentés par les flèches. D'abord, il choisit aléatoirement le nœud (6) pour échanger une partie de ses voisins (figure (a)). Ensuite, il envoie à ce nœud l'ensemble : $\{1, 4, 5\}$, et il reçoit simultanément de (6) : $\{6, 7, 9\}$. Puis les deux nœuds mettent à jour leurs listes de voisins (figure(b)). Par exemple, le nœud (1) a deux nouveaux voisins $\{7, 9\}$ remplacés par les nœuds $\{4, 6\}$. De la même façon, tous les autres nœuds appliquent le *PSS* et la topologie du réseau change à chaque itération.

Nous avons expliqué tout d'abord, notre technique pour la détection de modèles non fiables, puis notre algorithme d'agrégation, ainsi que le protocole de communication appliqué pour échanger les mo-

dèles entre les nœuds. Dans la suite, nous allons détailler notre critère de convergence, c'est-à-dire le critère d'arrêt de notre algorithme.

5.6 Critère de convergence

L'objectif de cette section est d'introduire un critère de convergence pour notre algorithme. Pour cela nous avons proposé une solution décentralisée, dans laquelle chaque nœud i applique soit une itération de l'algorithme, soit seulement le PSS. Le nœud i compare son modèle avec les modèles de ses voisins. Si son modèle est similaire à ceux de ses voisins, alors il applique seulement le PSS, sans exécuter les autres étapes de l'algorithme.

Le nœud i continue donc à faire tourner l'algorithme de PSS, même s'il atteint le modèle global. Notons que les nœuds ne peuvent pas savoir s'ils ont ou pas le modèle global. Ainsi, si le nœud i et ses nœuds voisins maintiennent le même modèle, cela ne signifie pas que le nœud i contienne le modèle global. Si à une itération donnée, il découvre un changement dans les modèles de ses voisins, il doit relancer l'algorithme d'agrégation. Le fait que l'algorithme ne s'arrête pas permet de gérer l'évolution du réseau (nœuds rejoignent ou quittent le réseau).

Plus précisément, considérons à une itération donnée, qu'un nœud i a obtenu son nouveau modèle M_i. Pour vérifier, si son modèle est similaire aux modèles de ses voisins, le nœud i itère entre les étapes suivantes :

- il ordonne ses voisins d'une façon croissante selon les valeurs de KL_{match} entre son modèle et les modèles de ses voisins ;
- il sélectionne le $percent$ième modèle M_{seuil} de sa liste ordonnée. Si ce modèle est comparable au modèle M_i, cela signifie que M_i est proche de tous les modèles fiables contenus dans sa liste de voisins ;
- il compare la divergence KL_{match} entre son modèle et le modèle M_{seuil} avec un seuil ϵ : si la valeur de KL_{match} est plus petite, alors le nœud i applique seulement le PSS, sinon une nouvelle itération de l'algorithme (10) est lancée. Notons que le paramètre ϵ est initialisé en pratique à une valeur très petite.

Le critère de convergence joue aussi un rôle important pour réduire la complexité de calcul de notre algorithme. Il évite en particulier d'exécuter les deux techniques principales (détection des modèles non fiables, agrégation des modèles), qui consomment la majorité du temps de calcul de l'algorithme principal (10).

5.7 Conclusion

Nous avons proposé une approche robuste et décentralisée pour estimer un modèle de mélange gaussien à partir de données distribuées. Elle consiste à fusionner dynamiquement les modèles locaux d'une façon robuste, c'est-à-dire en présence de modèles non fiables. L'agrégation des modèles est réalisée en utilisant seulement les paramètres des modèles. Les nœuds participants transmettent donc seulement les

paramètres, sans aucun accès aux données réelles.

Notre technique est spécifiée par plusieurs avantages :

– elle est caractérisée par une complexité de calcul réduite et un coût de transmission faible (seuls les paramètres sont utilisés) ;

– elle réalise une estimation robuste, en tenant compte de la présence des modèles non fiables ;

– elle est pertinente pour les réseaux dynamiques grâce à l'utilisation de PSS. Le PSS peut accélérer exponentiellement la convergence de l'algorithme [67, 13].

Cette technique présente beaucoup d'intérêts pour les systèmes collaboratifs décentralisés, par exemple pour le clustering de données ou les systèmes à base de recommandations.

Dans le chapitre suivant, nous allons valider cette technique par des expériences sur des données synthétiques et réelles. En particulier, nous allons tester les aspects principaux de notre proposition : la robustesse et la procédure d'agrégation.

CHAPITRE 6

Résultats expérimentaux de notre algorithme distribué

6.1 Introduction

Nous avons présenté dans le chapitre précédent, notre algorithme de clustering développé pour être appliqué dans des environnements distribués. Nous allons évaluer dans ce chapitre cette technique en effectuant des expériences d'abord sur des données synthétiques et ensuite sur des données réelles. L'objectif de ces expériences est de calculer le modèle global à partir des modèles générés sur des données réparties sur plusieurs nœuds du réseau. Ce modèle reflète la densité de probabilité de l'ensemble des données distribuées sur le réseau et il représente en même temps un clustering global, dû à l'utilisation de modèles de mélange. Nous allons examiner particulièrement dans ces expériences, les aspects principaux de notre technique comme la robustesse et la procédure d'agrégation.

Dans le reste de ce chapitre, nous présentons d'abord la façon d'évaluer un algorithme de clustering distribué. Nous montrons ensuite les propriétés générales du réseau dans laquelle notre technique est testée. Enfin nous présentons et discutons les résultats fournis par l'application de notre algorithme d'abord sur des données synthétiques, puis sur des données réelles.

6.2 Évaluation d'un algorithme de clustering distribué

Il existe très peu de mesures développées spécifiquement pour le clustering distribué. Par exemple, une technique simple peut être utilisée consistant à tester si les données se trouvent ou pas dans les mêmes clusters dans les deux clustering (centralisé et distribué). Ainsi, les résultats des algorithmes de clustering distribués peuvent être évalués par rapport à ceux obtenus par un algorithme de clustering centralisé appliqué sur toutes les données. Deux types de résultats peuvent être obtenus dans ce cas : exacts ou approximés.

– l'algorithme exact produit un modèle global identique au modèle hypothétique obtenu par un algorithme centralisé (c'est-à-dire sur l'ensemble des données). Des exemples produisant des résultats exacts ont été proposés dans [29, 125, 37, 55, 123].

– les algorithmes approximés produisent plutôt une approximation de modèle global qu'un modèle exact. Cette approximation est supposée être proche de celle générée par un clustering centralisé. La plupart des méthodes de clustering distribuées utilisent plutôt les algorithmes approximés. Elles réalisent des résultats comparables avec celles des algorithmes exacts, mais avec une complexité de calcul beaucoup moins élevée (calcul du modèle global à partir d'un nombre limité de données représentatives). De nombreux exemples d'algorithmes approximés sont proposés dans la littérature comme [56, 103, 65, 73, 52].

Dans notre algorithme, les résultats sont évalués en comparant à chaque itération les modèles locaux réalisés sur les différents nœuds avec le modèle global centralisé. Ce dernier modèle est calculé en supposant que tous les modèles sont localisés sur un seul site. Nous utilisons la divergence de KL pour mesurer la similarité entre les modèles locaux et le modèle centralisé. Nous allons voir qu'après certaines itérations le modèle global sera très comparable avec le modèle global centralisé (voir la section 6.3).

Dans la suite nous allons montrer les résultats de notre algorithme effectués sur des données synthétiques et réelles.

6.3 Résultats

Le but de cette section est d'évaluer notre algorithme de clustering, en particulier les aspects principaux : la robustesse et la procédure d'agrégation.

6.3.1 Propriétés générales du réseau

Nous présentons d'abord les différentes caractéristiques du réseau, sur lequel toutes les expériences ont été appliquées. En effet, il est composé par 200 nœuds. Chaque nœud du réseau est caractérisé par les propriétés suivantes :

– une liste de voisins composée de c nœuds (20 nœuds dans notre cas) ;

– un modèle de mélange gaussien. Ce modèle est généré à partir d'un algorithme de clustering probabiliste (*EM* dans ces expériences) ;

– un pourcentage (20%) de modèles non fiables parmis les modèles collectés de ses voisins (ce paramètre reflète le pourcentage des modèles non fiables dans le réseau).

Rappelons que chaque nœud échange une moitié $(c/2)$ de sa liste de voisins avec un autre nœud, lors de l'application de protocole *PSS*.

Après avoir présenté les caractéristiques des nœuds du réseau, nous montrerons dans la suite les résultats expérimentaux sur des données synthétiques.

6.3.2 Données synthétiques

L'objectif de cette expérience est d'évaluer en particulier la robustesse de notre algorithme distribué. Cet algorithme a été testé sur un ensemble de modèles supposés distribués sur les différents nœuds du réseau. Les modèles de nœuds sont générés comme suit :

- chaque nœud contient un mélange gaussien avec un nombre de composantes variant entre 3 et 12. Les moyennes de ses composantes sont générées aléatoirement selon la loi uniforme. En outre, les covariances sont initialisées d'une manière aléatoire, mais à condition qu'elles soient inversibles (non singulières). Le fait de choisir des matrices inversibles est expliqué par leur utilisation dans le calcul de divergence KL (il est nécessaire de calculer l'inverse des matrices pour trouver la divergence KL entre deux composantes gaussiennes).

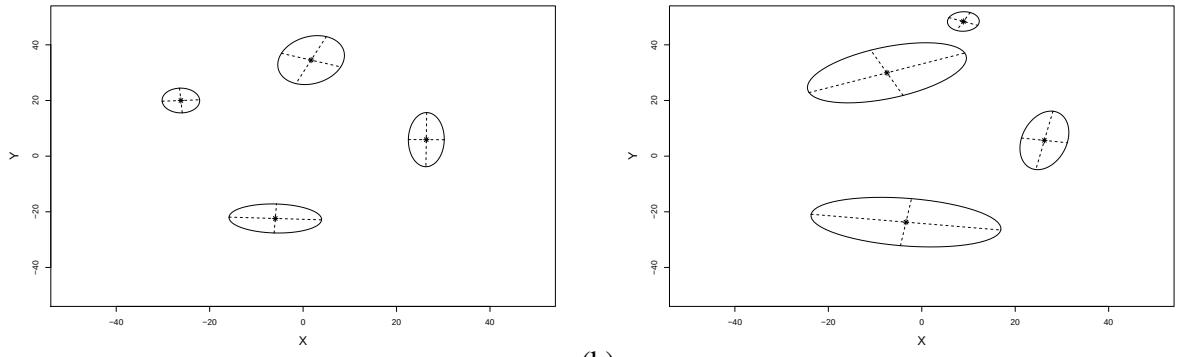
- les composantes de modèles non fiables sont générées aussi de la même manière, mais avec un nombre de composantes variant entre 1 et 3. Elles sont distribuées dans un espace de dimension plus large que celle correspondant aux modèles fiables. Celle-ci a pour but de réaliser des composantes s'écartant de celles de modèles fiables.

Rappelons que pour évaluer les résultats de notre algorithme de clustering, nous calculons le modèle global centralisé afin de le comparer avec les modèles locaux de nœuds obtenus durant chaque itération. Il est composé de 4 composantes représentant tous les modèles distribués dans le réseau. Le nombre de composantes est initialisé a priori. Le modèle global centralisé est obtenu à partir de l'ensemble de tous les modèles mais en supprimant avant les modèles non fiables (un pourcentage de 20% des modèles les plus éloignés a été enlevé parmi l'ensemble de modèles). Nous avons appliqué le même algorithme (11) pour éliminer les modèles non fiables. La figure (6.1(a)) illustre le modèle global centralisé (sans les modèles non fiables).

Pour montrer que notre technique de détection des modèles non fiables réussit à éliminer ce type de modèles, nous calculons aussi le modèle global centralisé en présence de modèles non fiables (voir figure (6.1(b)). Ce modèle représente une grande différence par rapport au modèle obtenu sans les modèles non fiables. En particulier les deux composantes situées à gauche dans la figure (6.1(b)). Elles sont plus larges que celles dans la figure (6.1(a)). Ainsi, la composante en haut est décalée en plus vers le haut et représente un volume plus petit. Les résultats obtenus montrent donc bien l'impact des modèles non fiables sur le processus d'agrégation.

La figure (6.2) illustre un exemple des modèles initiaux de 8 nœuds. Ils sont composés par un nombre de composantes différent et qui sont distribuées aléatoirement dans l'espace de données. Les nœuds 1, 2, 3, 4, 5, 6, 7 et 8 dans cette figure, contiennent respectivement des modèles avec 7, 9, 11, 2, 5, 8, 6 et 2 composantes. Les modèles des nœuds 1, 2, 3, 5, 6 et 7 sont des modèles fiables, tandis que les modèles des nœuds 4 et 8 sont considérés comme des modèles non fiables. Rappelons que ces derniers modèles sont composés par un petit nombre de composantes (entre 1 et 3) et que la distribution de leurs composantes est supposée loin par rapport à celles de modèles fiables.

La figure (6.3) montre la progression de notre algorithme. Elle représente les modèles de figure (6.2) après la première itération. Tous les modèles ont le même nombre (4) de composantes. Cependant, elles sont plus ou moins similaires au modèle global dans la figure (6.1(a)). Par exemple, les modèles 1, 2,



(a) (b)

Figure 6.1 – La figure (a) montre le modèle global centralisé obtenu après avoir éliminer les modèles non fiables, tandis que la figure (b) représente le modèle global en présence de modèles non fiables.

3, 5, 7 et 8 ont presque les mêmes formes et locations de composantes. Au contraire, les modèles de nœuds 4 et 6 présentent une grande différence par rapport à ce modèle (composantes à gauche et en bas). Celle-ci peut être expliquée par l'un de ces deux choix possibles :

- soit les voisins de ces nœuds contiennent des modèles non fiables avec un pourcentage plus grand que celui proposé a priori (c'est-à-dire plus que 20%). Dans ce cas, le processus d'agrégation est effectué sur un ensemble de modèles incluant des modèles non fiables, ce qui entraîne ainsi une déviation par rapport au modèle global réel ;

- soit les modèles à agréger ne contiennent pas de modèles non fiables, mais qui sont très différents par rapport au modèle global réel.

  La figure (6.5) représente l'évolution de notre critère de convergence tout au long des différentes itérations de notre algorithme. Le critère utilisé ici est la moyenne de la divergence de KL entre chaque modèle local et le modèle global réel. Remarquons que la courbe décroit directement après la première itération et tend directement vers zéro. Cette convergence est confirmée par les résultats obtenus dans la figure (6.4). Les modèles dans cette figure représentent ceux des nœuds dans la figure (6.2) après 13 itérations. Tous les modèles sont similaires entre eux et identiques au modèle global réel. Ces résultats montrent bien que notre technique de détection des modèles non fiables est efficace. Elle peut éliminer ce type de modèles en fournissant (dans la plupart du temps) au processus d'agrégation que les modèles fiables.

  Nous avons montré dans cette expérience que notre algorithme distribué peut trouver le modèle global réel. En particulier, nous avons évalué les aspects principaux de cet algorithme : la robustesse et la convergence. Dans la section suivante, nous présentons la deuxième expérience mais sur des données réelles.

6.3.3 Données réelles

La deuxième expérience est réalisée pour évaluer notre technique sur des données réelles. L'objectif de cette expérience est de calculer le modèle global à partir de modèles locaux estimés sur les données distribuées. Les données traitées dans cette expérience sont des données multimédias réelles. Elles représentent une collection de 406450 photos géolocalisées sur *Flikr*, et téléchargées par 5951 utilisateurs. La figure (6.6) montre la distribution de points (location des images) sur une carte géographique.

Pour tester notre algorithme de clustering sur cette collection de données réelles, nous l'avons partagé sur un réseau pair-à-pair composé par 200 nœuds, dans laquelle chaque nœud est défini par :

- un sous-ensemble de données présentées dans la figure (6.6). Le nombre de données dans chaque nœud varie entre 500 et 4000, et l'ensemble de ces données constitue les données locales de ce nœud ;

- un modèle de mélange gaussien. Ce modèle est estimé par l'algorithme de clustering EM, qui a été appliqué sur ses données locales. Pour déterminer la complexité de ce modèle (nombre de composantes), le critère BIC [106] a été utilisé pour tester tous les modèles avec des composantes entre 3 et 6. Les moyennes et les covariances des modèles ont été générées d'une façon aléatoire selon la loi uniforme, mais avec la condition que les matrices de covariances doivent être non singulières (inversibles).

Afin de comparer et d'évaluer la qualité des résultats de notre algorithme distribué, nous avons calculé le modèle global réel représentant l'ensemble des données. Pour cela, nous avons appliqué l'algorithme EM sur la collection de toutes les données. Nous avons déterminé la complexité de ce modèle par le critère de *BIC*. Le modèle optimisant ce critère est illustré dans la figure (6.6). Elle est composé par 5 composantes distinctes représentant le modèle global de toutes les données distribuées sur le réseau.

La figure (6.7) représente un exemple de modèles initiaux de 8 nœuds, obtenus par l'application de l'algorithme de clustering EM. Elle montre également la distribution de leurs points locaux. Les modèles présentés ici sont assez différents (nombre de composantes différent) possédant un ensemble local de données différent. Par exemple, les modèles des nœuds 1, 2, 3, 4, 5, 6, 7, et 8 dans cette figure, sont composés respectivement par 3, 5, 4, 4, 3, 5, 6 et 6 composantes. Notons aussi que par exemple, les modèles des nœuds 3 et 4 possèdent le même nombre de composantes (4 composantes chacun), mais chacun d'eux a une distribution différente (position et forme) de composantes.

Afin de montrer la progression de notre algorithme de clustering sur les données réelles, nous illustrons également dans la figure (6.8) les modèles de la figure (6.7) après la première itération, ainsi que leurs données locales. Tous les modèles dans cette figure, disposent du même nombre de composantes (5 dans notre cas). Ils commencent à être de plus en plus similaire au modèle global centralisé, même s'ils possèdent des représentations initiales différentes. Par exemple, les modèles initiaux des nœuds 3 et 8 dans la figure (6.7) sont très différents : chacun d'eux a un nombre différent de composantes (4 et 6 respectivement) qui ont des formes et positions distinctes. Ces modèles, après la première itération, deviennent très similaires et ils ressemblent aussi au modèle global centralisé. Cependant, les modèles des nœuds 1, 5 et 7 sont similaires entre eux, mais ils représentent une grande différence par rapport au modèle global centralisé (chacun d'eux a une composante qui est complètement déplacée de droite à

gauche). La similarité de chaque modèle par rapport au modèle global centralisé dépend directement des modèles des voisins de nœud contenant ce modèle. S'ils sont semblables alors leur agrégation réalise un modèle similaire au modèle global et inversement.

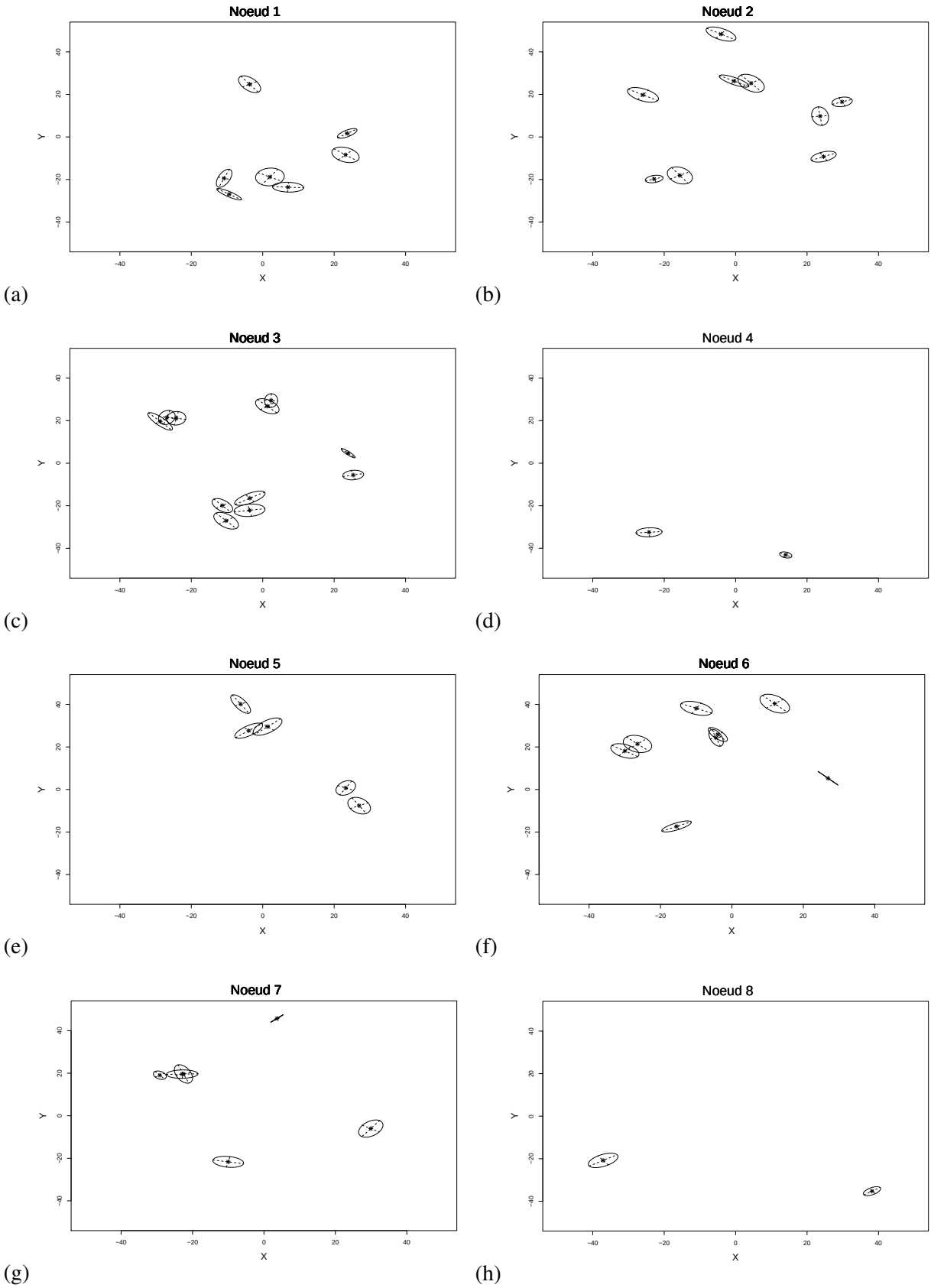
La figure (6.9) représente les 8 modèles du figure (6.7) après 10 itérations de notre algorithme de clustering. Tous les modèles obtenus sont similaires et identiques au modèle global centralisé dans la figure (6.6). Ils ont tous le même nombre de composantes et distribués de la même manière dans l'espace de dimension (forme et position de composantes). Pour montrer l'ajustement de notre modèle obtenu par rapport au jeu de données en étude, nous avons illustré également sur tous les nœuds la collection de toutes les données utilisées dans cette expérience. D'après les résultats obtenus dans cette figure, nous pouvons conclure que notre algorithme de clustering distribué réussit à converger, et que la propagation des modèles locaux selon le protocole de *gossip* semble efficace.

Afin d'évaluer la convergence de notre algorithme distribué, nous avons calculé la divergence KL entre chaque modèle local et le modèle globale centralisé. La figure (6.10) montre l'évolution de notre critère de convergence durant toutes les itérations de notre algorithme sur chaque nœud. La courbe décroît après chaque itération, et il converge rapidement vers zéro après la première itération. La convergence obtenue est bien confirmée par les résultats obtenus dans la figure (6.9) dans laquelle tous les modèles sont similaires et identiques au modèle global centralisé.

La comparaison entre la figure (6.6) et la figure (6.9) montre bien que les modèles obtenus sont similaires. Par conséquent, nous pouvons conclure que notre algorithme distribué réussit à estimer le modèle global correctement.

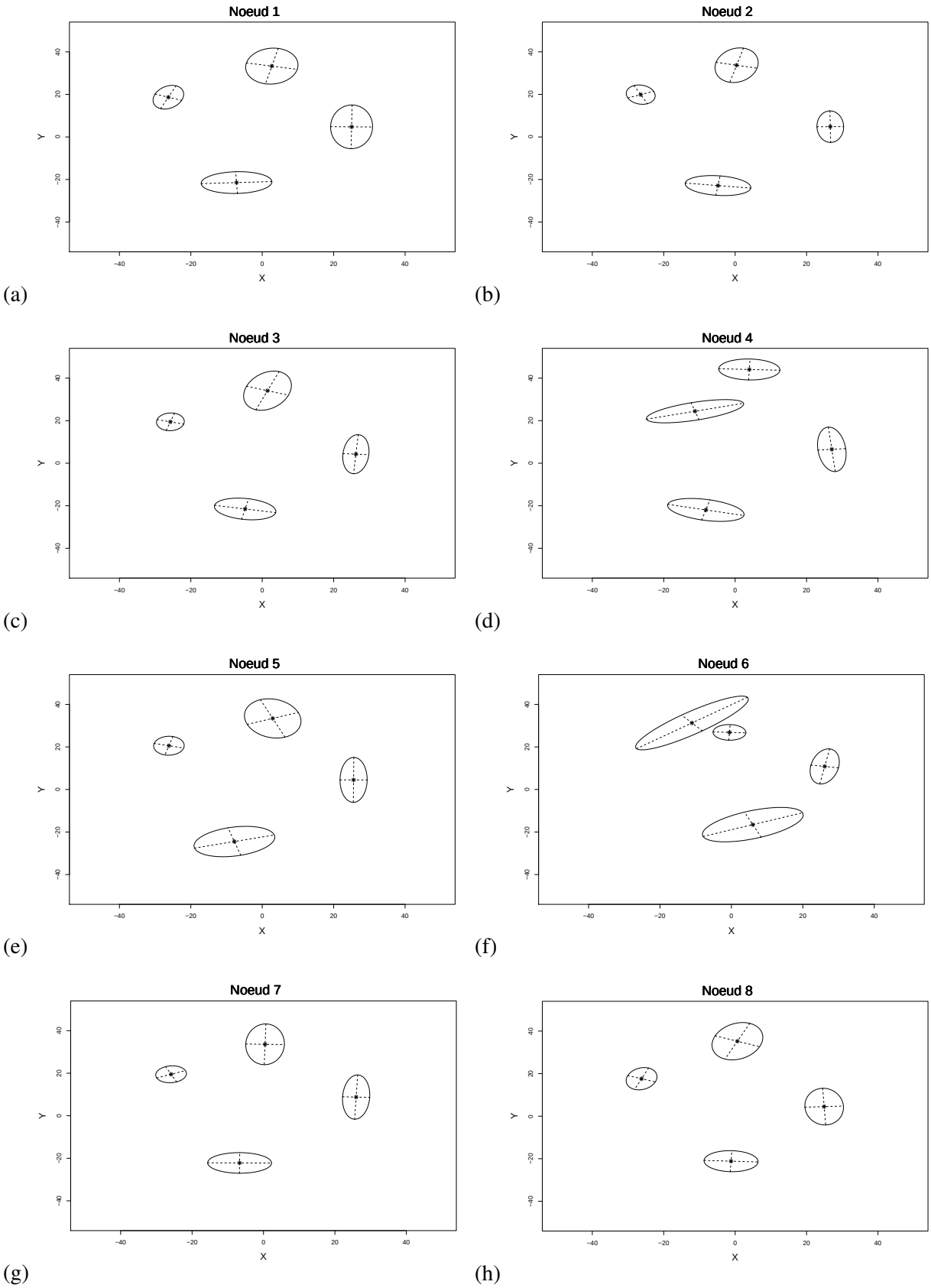
6.4 Conclusion

Nous avons présenté dans ce chapitre des expériences pour tester notre algorithme de clustering proposé dans le chapitre (5). Nous avons évalué en particulier les deux aspects principaux de cet algorithme : la robustesse et la procédure d'agrégation. Les expériences sont effectuées sur des données synthétiques et réelles. Les résultats obtenus montrent bien que notre algorithme distribué peut réaliser un modèle global identique à celui trouvé d'une manière centralisée. Il est donc considéré comme un algorithme efficace permettant à la fois d'identifier les modèles non fiables et d'obtenir une estimation correcte du modèle global.



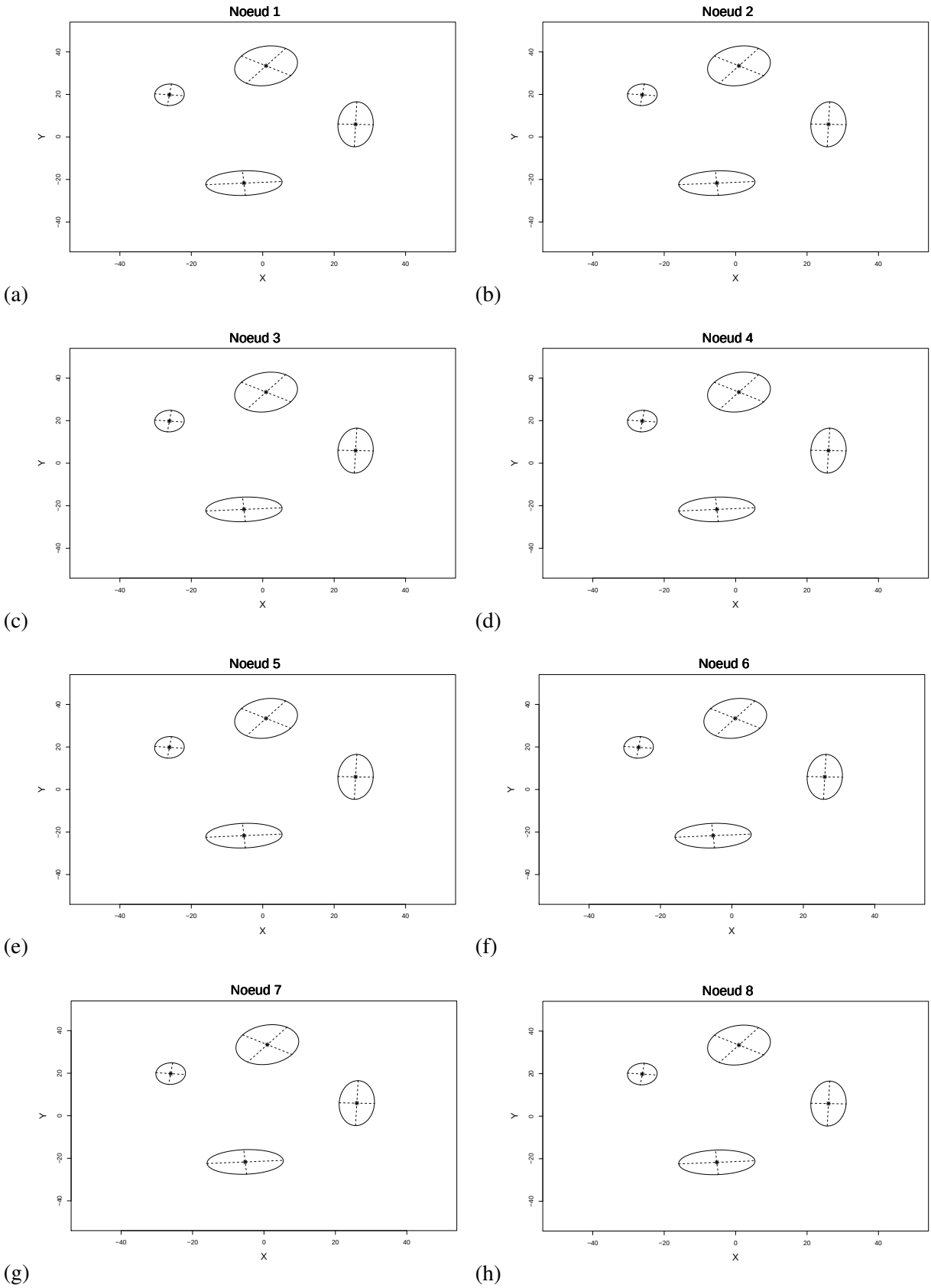
(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6.2 – Ces figures représentent les modèles initiaux de 8 nœuds d'un réseau pair-à-pair. Les figures (d) et (h) sont considérés comme des modèles non fiables.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6.3 – Ces figures représentent les modèles de 8 nœuds présentés dans la figure (6.2) après la première itération.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6.4 – Ces figures représentent les modèles de 8 nœuds présentés dans la figure (6.2) après 13 itérations.

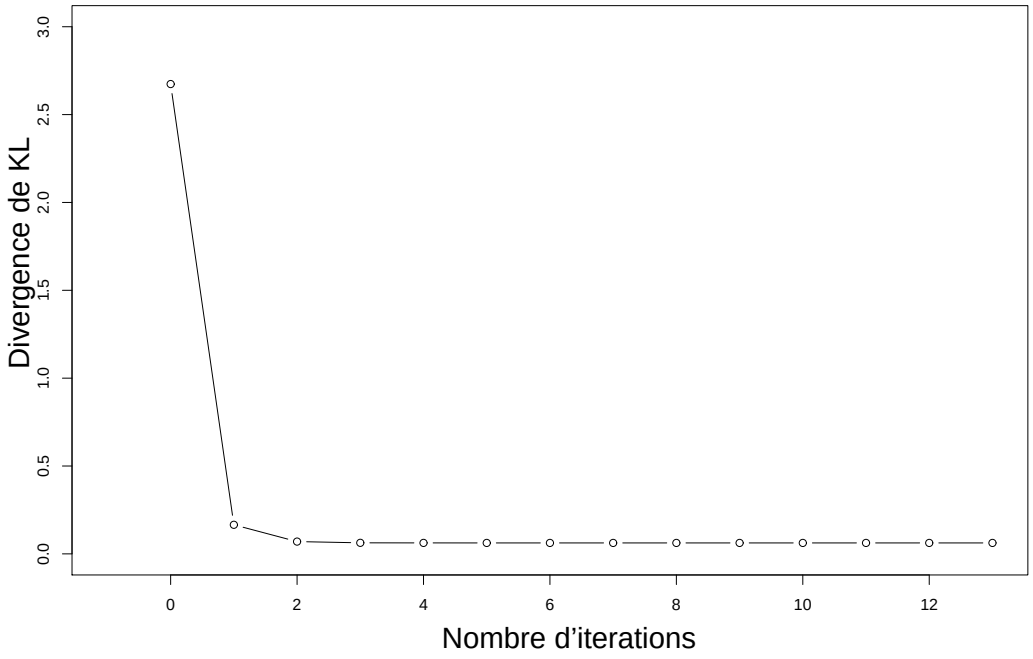


Figure 6.5 – Cette figure montre la moyenne de divergence KL entre le modèle de chaque nœud et le modèle global initial (sans modèles non fiables).

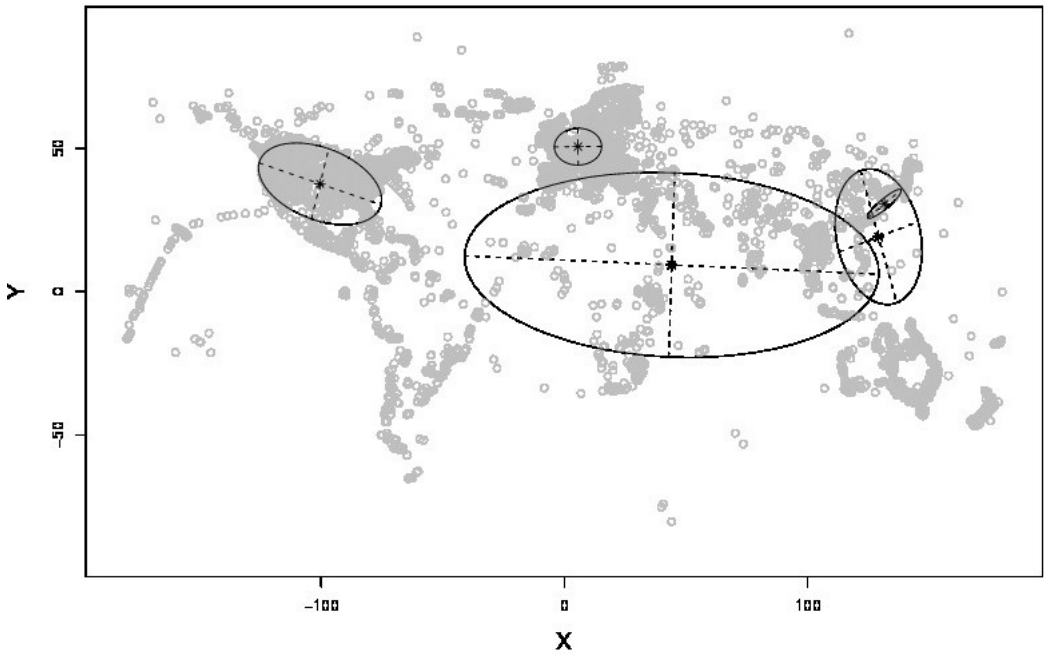
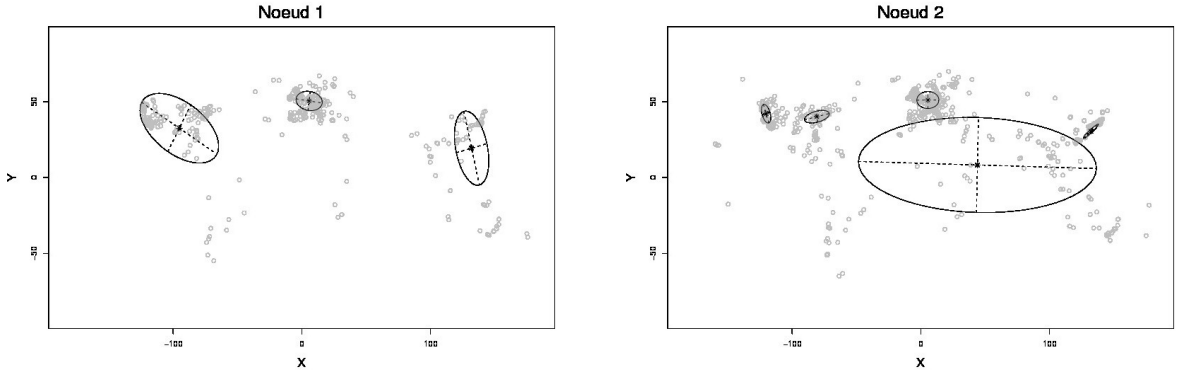
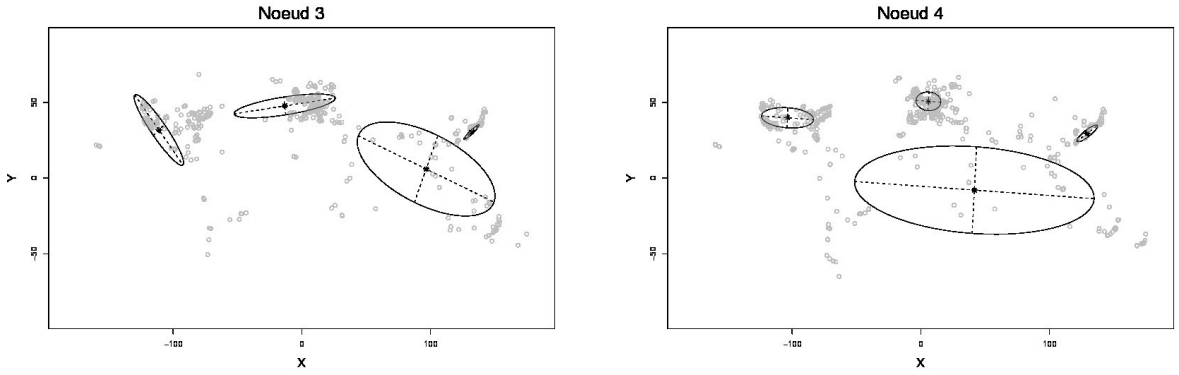


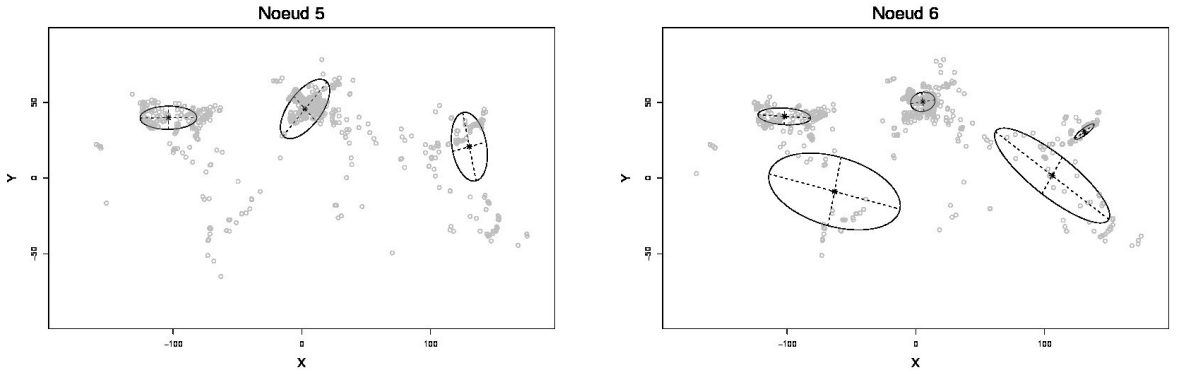
Figure 6.6 – Cette figure représente la localisation de toutes les images et le modèle global (estimé par EM) obtenu d'une manière centralisée.



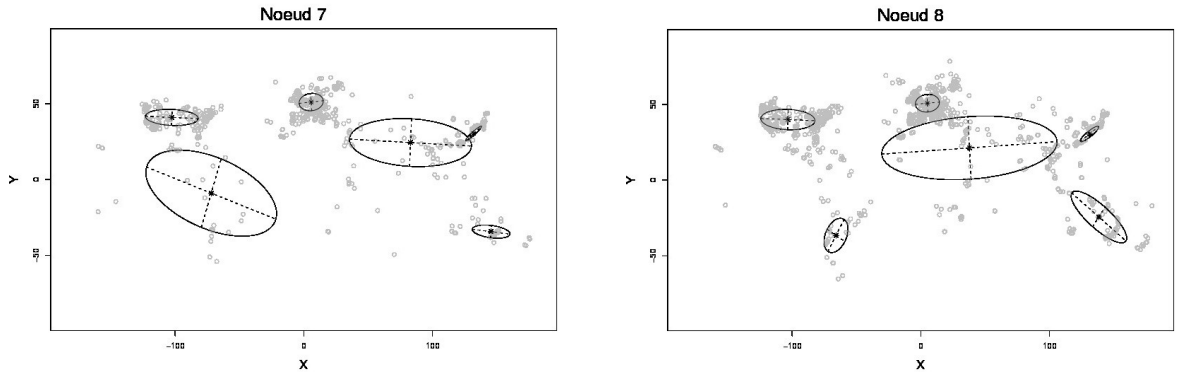
(a) (b)



(c) (d)

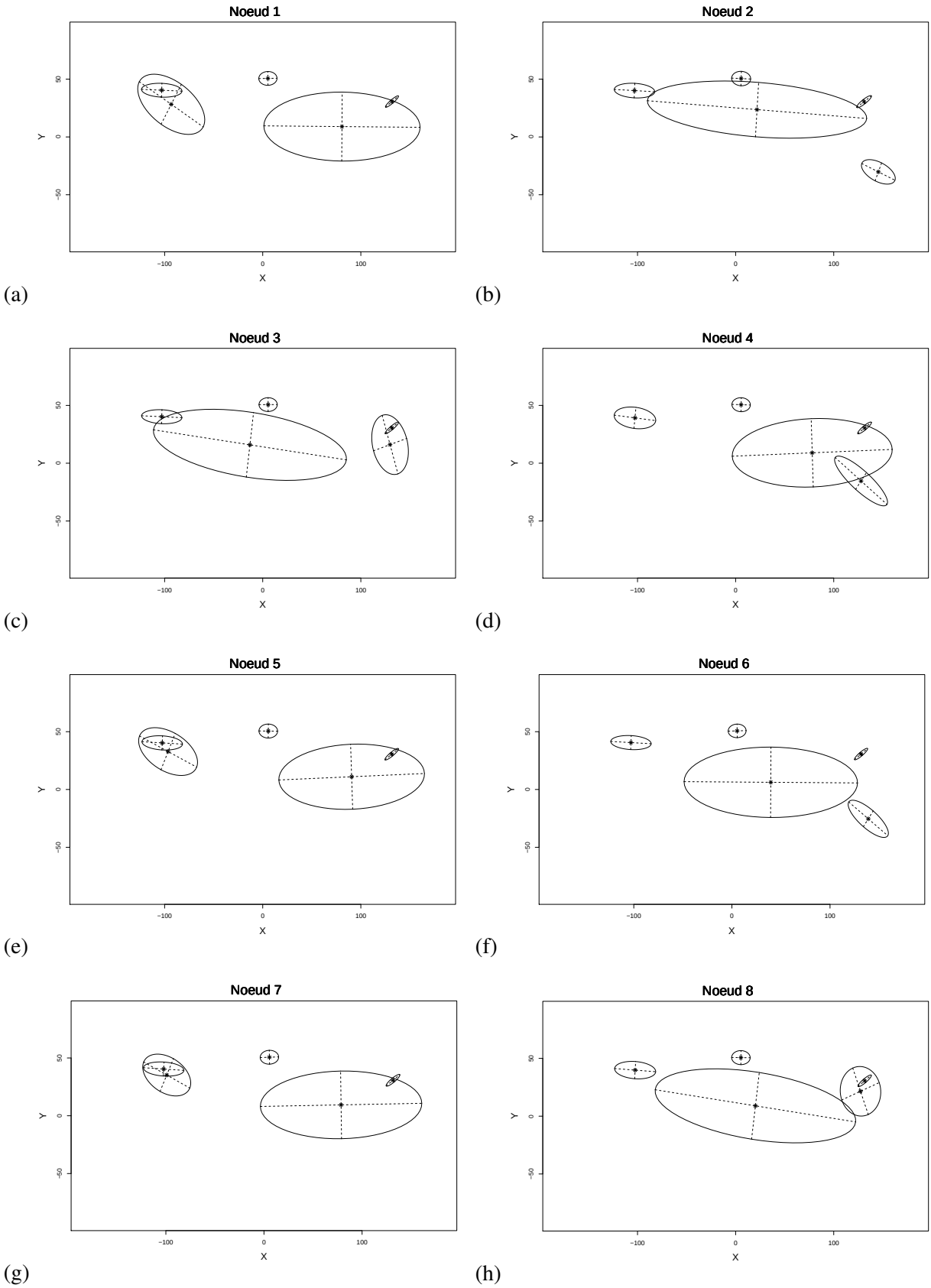


(e) (f)



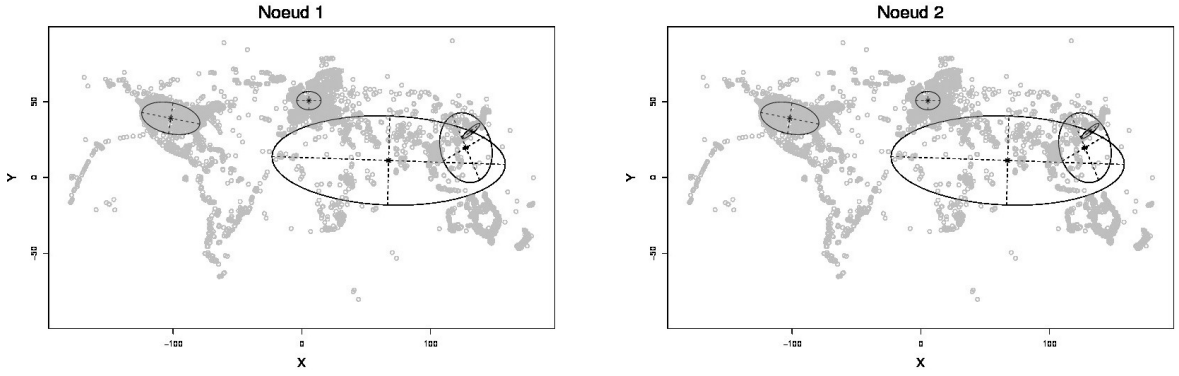
(g) (h)

Figure 6.7 – Ces figures représentent les modèles initiaux de 8 nœuds, après avoir appliqué localement l'algorithme EM sur leurs ensembles de données. Les points en gris représentent les données locales sur les nœuds.

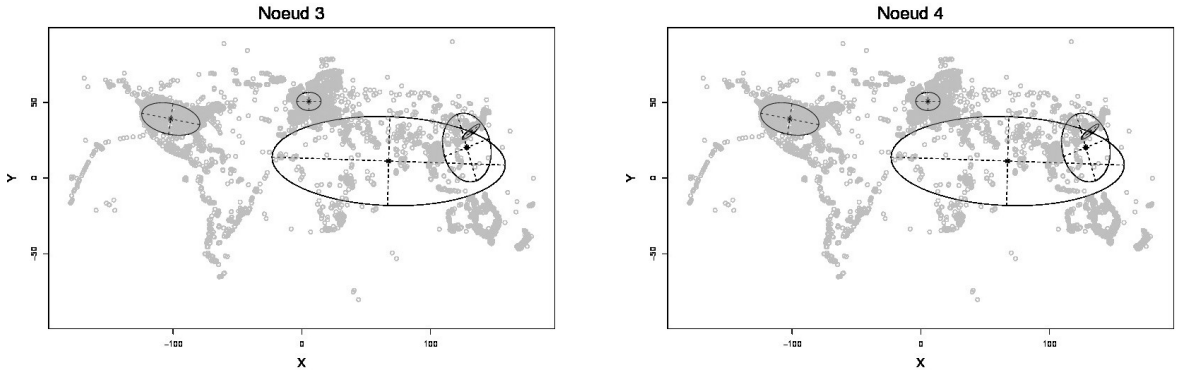


(a) (b) (c) (d) (e) (f) (g) (h)

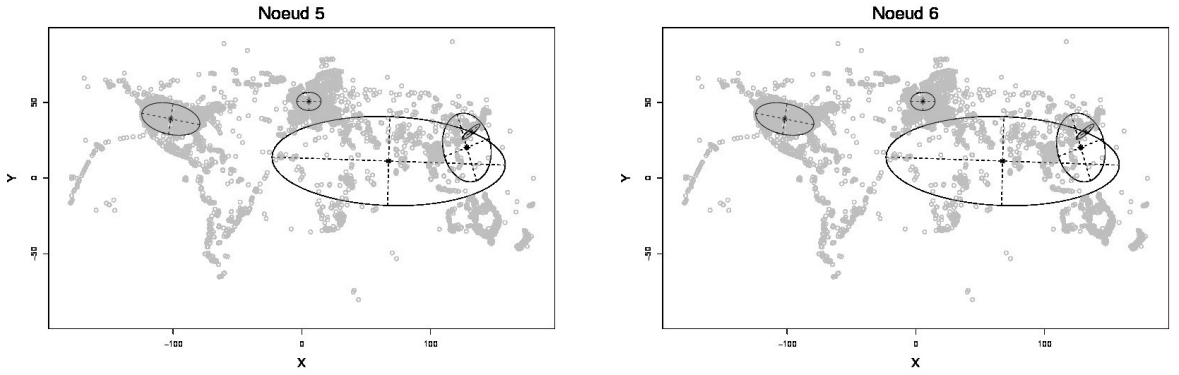
Figure 6.8 – Cette figure montre les modèles de 8 nœuds après exécuter une itération de notre algorithme principal.



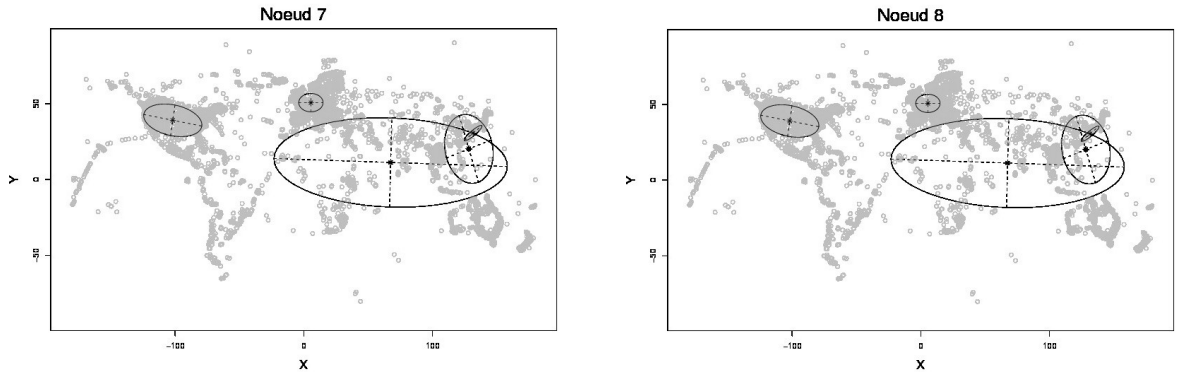
(a) (b)



(c) (d)



(e) (f)



(g) (h)

Figure 6.9 – Ces figures représentent les 8 modèles de la figure (6.7) après 10 itérations de notre algorithme d'estimation.

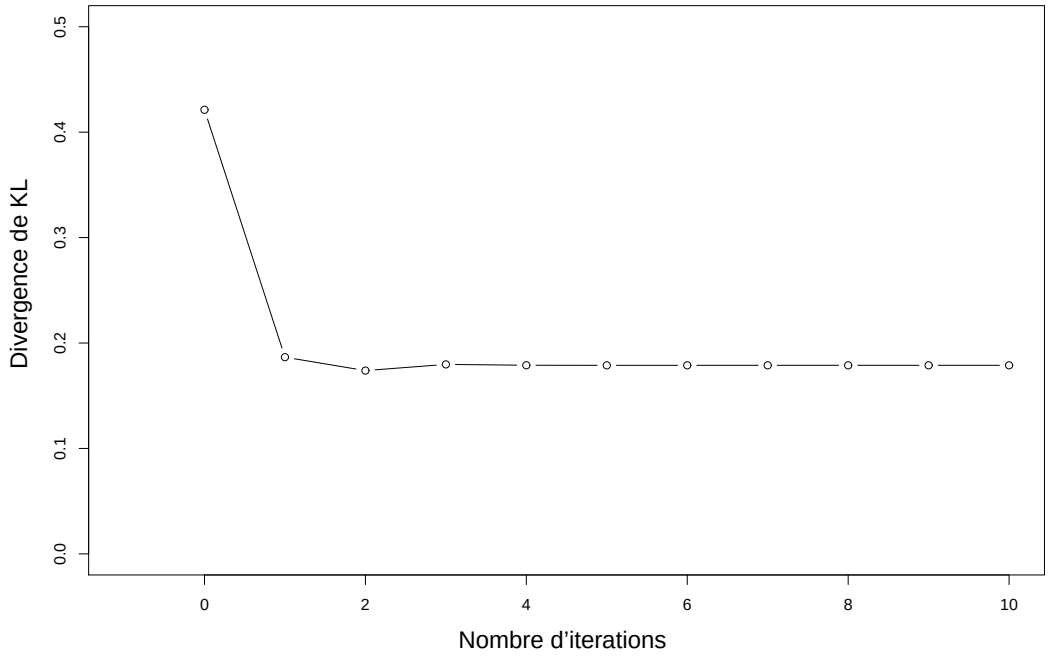


Figure 6.10 – Cette figure montre la moyenne de divergence de KL entre le modèle de chaque nœud et le modèle global centralisé.

CHAPITRE 7

Conclusion générale

Dans ce chapitre, nous allons décrire tout d'abord, les contributions principales de cette thèse. Puis, nous allons discuter quelques propositions pour le futur dans le domaine de l'apprentissage distribué.

7.1 Contributions principales

Nous avons abordé dans cette thèse, le problème d'apprentissage non supervisé en utilisant les modèles de mélange. Nous avons concentré notre étude sur l'agrégation de ce type de modèles, en proposant des techniques robustes, afin d'améliorer la qualité des résultats en présence des informations atypiques (données, modèles). Pour cela, une approche d'agrégation de modèles de mélange de *Student* a été proposée dans un premier temps [8]. Elle a pour but de réduire un modèle de mélange en un autre avec un nombre plus petit de composantes. Elle réalise une solution robuste due à l'utilisation de la distribution de *Student* (connue pour sa robustesse). Dans un deuxième temps, nous nous sommes dirigés vers l'apprentissage distribué, en particulier vers l'estimation des modèles de mélanges sur les données réparties. Nous avons proposé aussi une méthode d'agrégation robuste [10, 9], mais cette fois sur les mélanges gaussiens, en appliquant aussi une technique pour la détection des modèles non fiables.

Les solutions proposées dans notre thèse, sont caractérisées par plusieurs avantages :

- complexité de calcul : dans les deux méthodes proposées, l'agrégation est effectuée seulement sur les paramètres, sans aucun accès aux données réelles, ni sur des vecteurs d'attributs qui représentent ces données. En général, le nombre de paramètres est beaucoup moins nombreux que celui des données. Ceci entraîne une réduction de la complexité de calcul de nos algorithmes d'agrégation.

- coût de transmission : l'algorithme d'agrégation des modèles de mélange gaussien, est réalisé d'une façon distribuée. Les nœuds participants communiquent entre eux les paramètres des modèles, plutôt que les données. Ceci est dû à notre technique d'agrégation qui nécessite seulement l'utilisation des paramètres des modèles de mélange. Donc, la quantité des informations transmises dans le réseau est très faible.

- robustesse : le terme robustesse est considérée comme une propriété essentielle dans cette thèse. Elle est utilisée pour plusieurs raisons :

  - robustesse contre les informations atypiques (données, modèles) : les méthodes d'agrégation proposées sont effectuées d'une façon robuste, c'est-à-dire en tenant compte de la présence des

informations atypiques. Nous avons utilisé la distribution de *Student* dans la première méthode d'agrégation, pour faire face au problème des données atypiques. Dans la méthode d'agrégation de mélanges gaussiens, une technique robuste et décentralisée a été appliquée, pour identifier les modèles non fiables avant de réaliser la phase d'agrégation.

- robustesse du réseau : les nœuds dans notre proposition décentralisée, peuvent quitter ou rejoindre le réseau sans causer une dégradation importante sur l'estimation du modèle global. Ceci est dû à l'utilisation du protocole de communication *gossip*, qui est considéré aussi robuste dans le cas de défaillance des liens et des nœuds.

- respect d'anonymat et données privées : l'agrégation des modèles de mélange gaussien est effectuée seulement sur les paramètres de modèles, sans besoin de transmettre les données entre les nœuds. Chaque nœud peut donc garder ses données confidentielles, sans les partager ou donner l'accès à un autre nœud. Par conséquent, le problème de confidentialité est résolu implicitement.

Cette thèse est composée de quatre chapitres principaux, dont chacun d'eux aborde un problème spécifique. Dans le reste de cette section, nous concluons ces chapitres en de grandes lignes :

Nous avons commencé par proposer un état de l'art sur la technique de clustering non supervisé. Nous avons montré le concept et les propriétés essentielles de cette technique, ainsi que les principaux types de méthodes existantes. Ensuite, nous avons présenté d'une manière extensive les méthodes probabilistes, en introduisant dans un premier temps les modèles de mélange, et en expliquant dans un deuxième temps le principe de leur estimation. Aussi, nous avons étudié le problème des données atypiques. En effet, ce chapitre réalise une idée globale sur l'estimation des modèles de mélange, nécessaire pour introduire les autres chapitres.

Dans un deuxième temps, une méthode d'agrégation des modèles de mélange de *Student* a été utilisée afin de réduire un mélange de la distribution de *Student* en un autre, avec un nombre plus petit de composantes. Elle est robuste en présence des données atypiques, due à l'utilisation de la distribution de *Student*, qui peut modéliser ce type de données grâce à sa large queue. Elle se base seulement sur les paramètres de modèles pour réaliser la réduction. Elle est donc caractérisée par un temps de calcul beaucoup moins réduit que celles qui utilisent des données ou des vecteurs d'attributs décrivant ces données. Cette technique peut être aussi appliquée facilement sur des environnements distribués, en supposant par exemple que les modèles à agréger sont ceux des voisins de chaque élément du réseau. Cependant, elle nécessite dans ce cas, un protocole ou une stratégie pour recouvrir tous les éléments du réseau, et réaliser la synchronisation entre eux.

Nous avons abordé dans un troisième temps le problème du clustering distribué, c'est-à-dire sur des données supposées être réparties sur plusieurs sites. Tout d'abord, nous avons expliqué l'aspect général des approches de clustering distribuées. Ensuite, un état de l'art sur cette technique a été présenté, en créant en même temps plusieurs taxonomies selon les propriétés communes entre les différentes méthodes. Ainsi, nous avons étudié le problème des données atypiques dans le contexte de données distribuées. Ce chapitre a réalisé une étude globale sur la technique de clustering distribuée. Il a servi aussi à introduire notre proposition, détaillée dans le chapitre suivant, en particulier le protocole de *gossip* utilisé comme un protocole de communication dans notre technique proposée.

Dans un quatrième temps, nous avons proposé une technique d'estimation robuste des modèles de mélange gaussien, appliquée d'une façon décentralisée. En d'autres termes, elle consiste à fusionner dynamiquement les modèles locaux générés sur les différents sites, afin de construire un modèle global représentant toutes les données distribuées. L'estimation du modèle global est réalisée d'une façon robuste, en éliminant les modèles non fiables pouvant affecter les résultats. L'agrégation des modèles dans cette technique est effectuée seulement sur les paramètres des modèles. Donc, les nœuds participants transmettent seulement les paramètres entre eux, sans besoin de partager leurs données locales. Par conséquent, notre technique est caractérisée par un coût de transmission faible, aussi bien par une complexité de calcul réduite. Ainsi, le problème de confidentialité est implicitement résolu.

7.2 Perspectives

Dans cette thèse, nous avons proposé plusieurs techniques pour l'agrégation des modèles de mélanges sur des environnements distribués. Ce problème est envisagé de plus en plus récemment. Dans cette section, nous discutons certaines extensions de nos travaux pour le futur.

7.2.1 Application de la technique d'agrégation de modèles de *Student* sur des environnements distribués

Nous avons montré dans le chapitre (3) une technique d'agrégation des modèles de mélange de *Student*. Cette technique sert à réduire un mélange de *Student* en un autre avec un nombre plus petit de composantes. Il serait intéressant de tester cette approche sur des environnements distribués. Nous pouvons appliquer le même algorithme (10) présenté dans le chapitre (5) mais sur des mélanges de *Student* au lieu des gaussiennes. La technique de détection des modèles non fiables reste la même en utilisant la divergence de KL entre les mélanges de *Student*. Nous pouvons aussi garder le même service (*PSS*) pour changer la topologie du réseau. Cette proposition sera l'un de nos prochains travaux dans le futur.

7.2.2 Amélioration de nos propositions

Nous avons présenté dans le chapitre (5) une technique d'estimation robuste de modèles de mélange gaussien. Nous décrivons dans cette section plusieurs propositions pour l'améliorer et résoudre certains problèmes.

– **détermination automatique de nombre de composantes du modèle global :** rappelons que ce nombre est considéré comme un paramètre critique de nos propositions. Il est aussi intéressant de laisser l'algorithme trouver le nombre correct de composantes. Ce problème peut être résolu par des solutions classiques. Par exemple, nous pouvons appliquer la technique d'agrégation sur chaque nœud avec un nombre de composantes (modèle réduit) variant entre le nombre minimal et maximal de composantes de mélange des voisins. Nous pouvons choisir ensuite le modèle qui minimise la divergence avec l'ensemble des modèles de voisins. Une autre solution peut aussi être proposée sur chaque nœud. Celle-ci consiste à générer un échantillon de points du modèle moyen (somme pondérée des modèles des voisins de ce nœud) [80]. Ensuite un algorithme comme *EM*

peut être appliqué avec le critère *BIC* pour trouver le nombre optimal des composantes.

- **utilisation des composantes de modèles au lieu du modèle lui-même :** il est peut-être intéressant de travailler sur des composantes de modèles plutôt que sur les modèles eux-mêmes, afin d'éliminer les modèles non fiables. Un modèle peut contenir à la fois des composantes fiables (représentant les données typiques) et non fiables (représentant les données atypiques). Ce modèle peut être considéré comme un modèle non fiable et celui-ci peut entraîner de ne pas choisir les composantes fiables (elles peuvent être des composantes représentatives). Il est donc plus pertinent de supprimer les composantes non fiables plutôt que d'éliminer toutes les composantes du modèle. Nous pouvons dans ce cas, choisir par exemple pour chaque composante ses K plus proches voisins et de lui attribuer un score d'atypicité (comme la moyenne de sa distance avec ses K plus proches voisins). Cette technique peut remplacer celle proposée dans notre algorithme pour identifier les modèles non fiables.

- **une autre technique pour la diffusion des informations autre que *PSS* :** nous avons utilisé dans notre algorithme d'estimation décentralisée la service *PSS* pour accélérer la diffusion des informations dans le réseau et par conséquent la convergence de l'algorithme. Elle est aussi pertinente pour les réseaux dynamiques. Cependant elle augmente la complexité de calcul de l'algorithme principal (10) en ajoutant une étape supplémentaire (échange des voisins entre les nœuds). Elle impose des contraintes sur le nombre de voisins des nœuds (même valeur pour tous les nœuds, qui doit être non petite). Il est intéressant de tester notre algorithme sans aucune contrainte sur le réseau (c'est-à-dire que les nœuds peuvent avoir un nombre différent des voisins qui peut être aussi petit). Nous pouvons appliquer par exemple la technique proposée dans [23] pour diffuser l'information.

7.2.3 Application de nos propositions sur les autres mélanges de la famille exponentielle

Nous avons vu que nos techniques d'agrégation se basent directement sur la similarité entre des modèles de mélange. Nous avons montré comment est calculée la divergence entre les mélanges gaussiens, ainsi qu'entre les mélanges de *Student*. Rappelons que nous avons utilisé la divergence de KL comme une mesure de similarité entre ces deux genres de mélange. Notons qu'une fois, une similarité entre deux autres mélanges soit définie, alors nous pouvons appliquer facilement nos approches. Les auteurs de [42] ont montré que la divergence de *Bregman* peut être appliquée sur toutes les distributions de la famille exponentielle. Mais les modèles de mélange ne font en général pas partie de la famille exponentielle. Il est donc peut-être intéressant de déterminer une mesure de similarité (par exemple KL) entre les autres mélanges de la famille exponentielle (Laplacian, Poisson, Binomial, Bernoulli, etc) pour appliquer nos techniques.

Bibliographie

[1] O. A. ABBAS : Comparisons between data clustering algorithms. *The International Arab Journal of Information Technology*, 5(3):320–325, 2008.

[2] G. ADOMAVICIUS et A. TUZHILIN : Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[3] R. AGRAWAL, J. GEHRKE, D. GUNOPULOS et P. RAGHAVAN : Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Record*, 27(2):94–105, 1998.

[4] A. AHMED et A. SMOLA : Www 2011 invited tutorial overview : latent variable models on the internet. *In Proceedings of the 20th international conference companion on World wide web*, p. 281–282. ACM, 2011.

[5] F. ANGIULLI et C. PIZZUTI : Fast outlier detection in high dimensional spaces. p. 15–26, 2002.

[6] M. ANKERST, M. M. BREUNIG, H. P. KRIEGEL et J. SANDER : Optics : Ordering points to identify the clustering structure. *In SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data*, p. 49–60, 1999.

[7] D. ARTHUR et S. VASSILVITSKII : k-means++ : the advantages of careful seeding. *In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027–1035, 2007.

[8] A. E. ATTAR, A. PIGEAU et M. GELGON : Fast aggregation of student mixture m els. *In European Signal Processing Conference (Eusipco'2009)*, p. 312–216, 2009.

[9] A. E. ATTAR, A. PIGEAU et M. GELGON : A decentralized and robust approach to estimating a probabilistic mixture model for structuring distributed data. *In Proceedings of the 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, p. 372–379, 2011.

[10] A. E. ATTAR, A. PIGEAU et M. GELGON : A decentralized technique for robust probabilistic mixture modelling of a distributed data set. *In Intelligent Distributed Computing V - Proceedings of the 5th International Symposium on Intelligent Distributed Computing*, p. 271–276, 2011.

[11] G. BALL et D. HALL : A clustering technique for summarizing multivariate data. *Behaviorial Sciences*, 12(2):153–155, 1967.

[12] P. BERKHIN : A survey of clustering data mining techniques. *In Grouping Multidimensional Data*, p. 25–71. 2006.

[13] K. BHADURI et A. N. SRIVASTAVA : A local scalable distributed expectation maximization algorithm for large peer-to-peer networks. *In The Ninth IEEE International Conference on Data Mining*, p. 31–40, 2009.

[14] N. BICOCCHI, M. MAMEI et F. ZAMBONELLI : Handling dynamics in gossip-based aggregation schemes. *In Proceedings of the 14th IEEE Symposium on Computers and Communications*, p. 380–385, 2009.

[15] C. M. BISHOP : *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

[16] J. W. BRANCH, C. GIANNELLA, B. K. SZYMANSKI, R. WOLFF et H. KARGUPTA : In-network outlier detection in wireless sensor networks. *In Distributed Computing Systems*, p. 51–59, 2006.

[17] M. M. BREUNIG, H. P. KRIEGEL, R. T. NG et J. SANDER : Lof : Identifying density-based local outliers. *In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, p. 93–104, 2000.

[18] P. BRUNEAU, M. GELGON et F. PICAROUGNE : Aggregation of probabilistic pca mixtures with a variational-bayes technique over parameters. *In 20th International Conference on Pattern Recognition*, p. 702–705, 2010.

[19] P. BRUNEAU, M. GELGON et F. PICAROUGNE : Parsimonious reduction of gaussian mixture models with a variational-bayes approach. *Pattern Recognition*, 43(3):850–858, 2010.

[20] C. CARSON, S. BELONGIE, H. GREENSPAN et J. MALIK : Blobworld : Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, 1999.

[21] G. CELEUX et G. GOVAERT : A classification em algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, 14(3):315–332, 1992.

[22] V. CHANDOLA, A. BANERJEE et V. KUMAR : Outlier detection : A survey. Rap. tech., Department of Computer Science and Engineering University of Minnesota, 2007.

[23] S. DATTA, C. GIANNELLA et H. KARGUPTA : Approximate distributed k-means clustering over a peer-to-peer network. *IEEE Transactions on Knowledge and Data Engineering*, 21(10):1372–1388, 2009.

[24] R. DAVE et S. SEN : On generalizing the noise clustering algorithms. *In Proceedings of the 7th International Fuzzy Systems Association World Congress*, p. 205–210, 1997.

[25] R. N. DAVE : Characterization and detection of noise in clustering. *Pattern Recognition Letters*, 12(11):657–664, 1991.

[26] R. N. DAVE et R. KRISHNAPURAM : Robust clustering methods : a unified view. *IEEE Transactions on Fuzzy Systems*, 5(2):270–293, 1997.

[27] D. DEB, M. M. FUAD et R. A. ANGRYK : Distributed hierarchical document clustering. *In IASTED International Conference on Advances in Computer Science and Technology*, p. 328–333, 2006.

[28] A. P. DEMPSTER, N. M. LAIRD et D. B. RUBIN : Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[29] I. DHILLON et D. MODHA : A data-clustering algorithm on distributed memory multiprocessors. *In In Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence*, p. 245–260, 2000.

[30] M. EISENHARDT, W. MÜLLER et A. HENRICH : Classifying documents by distributed p2p clustering. *In INFORMATIK 2003 - Innovative Informatikanwendungen, Band 2, Beiträge der 33. Jahrestagung der Gesellschaft für Informatik*, p. 286–291, 2003.

[31] E. ESKIN, A. ARNOLD, M. PRERAU, L. PORTNOY et S. STOLFO : A geometric framework for unsupervised anomaly detection : Detecting intrusions in unlabeled data. *In Applications of Data Mining in Computer Security*, p. 77–102, 2002.

[32] M. ESTER, H. P. KRIEGEL, J. SANDER et X. XU : A Density-Based algorithm for discovering clusters in large spatial databases with noise. *In Second International Conference on Knowledge Discovery and Data Mining*, p. 226–231, 1996.

[33] H. T. F. KLAWONN, R. Kruse : Fuzzy shell cluster analysis. *In Learning, Networks and Statistics*, p. 105–120, 1997.

[34] Y. FERNANDESS, A. FERNÁNDEZ et M. MONOD : A generic theoretical framework for modeling gossip-based algorithms. *Operating Systems Review*, 41(5):19–27, 2007.

[35] P. A. FORERO, A. CANO et G. B. GIANNAKIS : Consensus-based distributed expectation-maximization algorithm for density estimation and classification using wireless sensor networks. *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, p. 1989–1992, 2008.

[36] G. FORESTIER : *Connaissances et clustering collaboratif d'objets complexes multisources*. Thèse de doctorat, Université de Strasbourg, 2010.

[37] G. FORMAN et B. ZHANG : Distributed data clustering can be efficient and exact. *ACM SIGKDD explorations newsletter*, 2(2):34–38, 2000.

[38] C. FRALEY et A. E. RAFTERY : How many clusters ? which clustering method ? answers via Model-Based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

[39] C. FRALEY et A. E. RAFTERY : Minimum volume ellipsoid. *Wiley Interdisciplinary Reviews : Computational Statistics*, 1(1):71–82, 2009.

[40] A. L. N. FRED et A. K. JAIN : Combining multiple clusterings using evidence accumulation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 27(6):835–850, 2005.

[41] L. A. GARCÍA-ESCUDERO, A. GORDALIZA, C. MATRÁN et A. MAYO-ISCAR : A general trimming approach to robust cluster analysis. *The Annals of Statistics*, 36(3):1324–1345, 2008.

[42] V. GARCIA et F. NIELSEN : Simplification and hierarchical representations of mixtures of exponential families. *Signal Processing*, 90(12):3197–3212, 2010.

[43] V. GARCIA, F. NIELSEN et R. NOCK : Hierarchical gaussian mixture model. *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, p. 4070–4073, 2010.

[44] L. A. GARCÍA-ESCUDERO, A. GORDALIZA, C. MATRÁN et A. MAYO-ISCAR : A review of robust clustering methods. *Adv. Data Analysis and Classification*, 4(2):89–109, 2010.

[45] J. GOLDBERGER, S. GORDON et H. GREENSPAN : An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. *In 9th IEEE International Conference on Computer Vision*, p. 487–493, 2003.

[46] J. GOLDBERGER, H. GREENSPAN et J. DREYFUSS : Simplifying mixture models using the unscented transform. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8):1496–1502, 2008.

[47] J. GOLDBERGER et S. ROWEIS : Hierarchical clustering of a mixture model. *In Advances in Neural Information Processing Systems 17*, p. 505–512, 2005.

[48] N. GRIRA, M. CRUCIANU et N. BOUJEMAA : Unsupervised and semi-supervised clustering : a brief survey. A Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence, 2004.

[49] D. GU : Distributed em algorithm for gaussian mixtures in sensor networks. *IEEE Transactions on Neural Networks*, 19(7):1154–1166, 2008.

[50] S. GUHA, R. RASTOGI et K. SHIM : Cure : An efficient clustering algorithm for large databases. *In SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*, p. 73–84, 1998.

[51] S. GUHA, R. RASTOGI et K. SHIM : Rock : A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000.

[52] K. GUO et Z. LIU : A new efficient hierarchical distributed p2p clustering algorithm. *In Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, p. 352–355, 2008.

[53] P. HAASE, D. HERZIG, M. A. MUSEN et T. TRAN : Semantic wiki search. *In The Semantic Web : Research and Applications, 6th European Semantic Web Conference*, p. 445–460, 2009.

[54] K. M. HAMMOUDA : *Distributed document clustering and cluster summarization in peer-to-peer environments*. Thèse de doctorat, University of Waterloo, Department of Electrical and Computer Engineering, 2007.

[55] K. M. HAMMOUDA et M. S. KAMEL : Collaborative document clustering. *In Proceedings of the Sixth SIAM International Conference on Data Mining*, p. 453–463, 2006.

[56] K. M. HAMMOUDA et M. S. KAMEL : Hierarchically distributed peer-to-peer document clustering and cluster summarization. *Knowledge and Data Engineering, IEEE Transactions on*, 21(5):681–698, 2009.

[57] J. HARDIN et D. M. ROCKE : Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator. *Computational Statistics & Data Analysis*, 44(4):625–638, 2004.

[58] Z. HE, X. XU et S. DENG : Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003.

[59] J. R. HERSHEY et P. A. OLSEN : Approximating the kullback leibler divergence between gaussian mixture models. *In Acoustics, Speech and Signal Processing*, p. 317–320, 2007.

[60] A. HINNEBURG, E. HINNEBURG et D. A. KEIM : An efficient approach to clustering in large multimedia databases with noise. *In in Proceedings of 4th International Conference in Knowledge Discovery and Data Mining*, p. 58–65, 1998.

[61] V. HODGE et J. AUSTIN : A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[62] Z. HUANG : Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[63] A. K. JAIN : Data clustering : 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[64] A. K. JAIN, M. N. MURTY et P. J. FLYNN : Data clustering : A review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[65] E. JANUZAJ, H. P. KRIEGEL et M. PFEIFLE : Dbdc : Density based distributed clustering. *In Advances in Database Technology EDBT*, p. 88–105, 2004.

[66] M. JELASITY et O. BABAOGLU : T-man : Gossip-based overlay topology management. *In Engineering Self-Organising Systems, Third International Workshop*, p. 1–15, 2005.

[67] M. JELASITY, A. MONTRESOR et O. BABAOGLU : Gossip-based aggregation in large dynamic networks. *ACM Transaction on Computer Systems*, 23(3):219–252, 2005.

[68] M. JELASITY, S. VOULGARIS, R. GUERRAOUI, A. M. KERMARREC et M. van STEEN : Gossip-based peer sampling. *ACM Transactions Computer Systems*, 25(3):1–36, 2007.

[69] E. L. JOHNSON et H. KARGUPTA : Collective, hierarchical clustering from distributed, heterogeneous data. *In Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems*, p. 221–244, 2000.

[70] S. JULIER et J. K. UHLMANN : A general method for approximating nonlinear transformations of probability distributions. Rap. tech., Department of Engineering Science, University of Oxford, 1996.

[71] H. KARGUPTA, W. HUANG, K. SIVAKUMAR et E. L. JOHNSON : Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.

[72] G. KARYPIS, E. H. HAN et V. KUMAR : Chameleon : Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[73] R. KASHEF : *Cooperative Clustering Model and Its Applications*. Thèse de doctorat, University of Waterloo, Department of Electrical and Computer Engineering, 2008.

[74] S. R. KASHYAP, S. DEB, K. V. M. NAIDU, R. RASTOGI et A. SRINIVASAN : Efficient gossip-based aggregate computation. *In Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, p. 308–317, 2006.

[75] A. M. KERMARREC et M. van STEEN : Gossiping in distributed systems. *Operating Systems Review*, 41(5):2–7, 2007.

[76] E. M. KNORR, R. T. NG et V. TUCAKOV : Distance-Based outliers : Algorithms and applications. *VLDB Journal : Very Large Data Bases*, 8(3):237–253, 2000.

[77] W. KOWALCZYK et N. VLASSIS : Newscast em. *In Advances in Neural Information Processing Systems 17*, p. 713–720. 2005.

[78] C. LIU et D. RUBIN : Ml estimation of the t distribution using em and its extensions, ecm and ecme. *Statistica Sinica*, 5(1):19–39, 1995.

[79] M. D. M. HUBERT : Minimum covariance determinant. *Wiley Interdisciplinary Reviews : Computational Statistics*, 2(1):36–43, 2010.

[80] S. MERUGU et J. GHOSH : Privacy-preserving distributed clustering using generative models. *In Proceedings of the Third IEEE International Conference on Data Mining*, p. 211–218, 2003.

[81] E. MONTIJANO, S. MARTÍNEZ et S. SAGUÉS : De-ransac : Robust distributed consensus in sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics : part B*, 2010.

[82] A. MONTRESOR : Intelligent gossip. *In Intelligent Distributed Computing, Systems and Applications, Proceedings of the 2nd International Symposium on Intelligent Distributed Computing*, p. 3–10, 2008.

[83] R. M. NEAL et G. E. HINTON : A view of the em algorithm that justifies incremental, sparse, and other variants. *In Proceedings of the NATO Advanced Study Institute on Learning in graphical models*, p. 355–368, 1998.

[84] R. T. NG et J. HAN : Clarans : A method for clustering objects for spatial data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 14(5):1003–1016, 2002.

[85] S. K. NG et G. J. MCLACHLAN : Robust estimation in gaussian mixtures using multiresolution kd-trees. *In Proceedings of the Seventh International Conference on Digital Image Computing : Techniques and Applications*, p. 145–154, 2003.

[86] F. NIELSEN, V. GARCIA et R. NOCK : Gaussian mixture models via entropic quantization. *In European Signal Processing Conference (Eusipco'2009)*, p. 2012–2016, 2009.

[87] F. NIELSEN et R. NOCK : Sided and symmetrized bregman centroids. *Information Theory, IEEE Transactions on*, 55(6):2882–2904, 2009.

[88] R. D. NOWAK : Distributed em algorithms for density estimation and clustering in sensor networks. *IEEE Transactions on Signal Processing*, 51(8):2245–2253, 2003.

[89] J. H. OH et J. GAO : A kernel-based approach for detecting outliers of high-dimensional biological data. *BMC Bioinformatics*, 10(S-4), 2009.

[90] P. A. OLSEN et S. DHARANIPRAGADA : An efficient integrated gender detection scheme and time mediated averaging of gender dependent acoustic models. *In 8th European Conference on Speech Communication and Technology*, p. 2509–2512, 2003.

[91] T. PALPANAS, D. PAPADOPOULOS, V. KALOGERAKI et D. GUNOPULOS : Distributed deviation detection in sensor networks. *SIGMOD Record*, 32(4):77–82, 2003.

[92] S. PATTERSON, B. BAMIEH et A. E. ABBADI : Distributed average consensus with stochastic communication failures. *In Proceedings of the 46th IEEE Conference on Decision and Control*, p. 490–495, 2007.

[93] D. PEEL et G. J. MCLACHLAN : Robust mixture modelling using the t distribution. *Statistics and Computing*, 10(4):339–348, 2000.

[94] A. PIGEAU : *Structuration géo-temporelle de données multimédia personnelles en vue de la navigation sur un appareil mobile*. Thèse de doctorat, Université de Nantes, 2005.

[95] S. RAJASEGARAR, C. LECKIE, M. PALANISWAMI et J. C. BEZDEK : Quarter sphere based distributed anomaly detection in wireless sensor networks. p. 3864–3869, 2007.

[96] B. RAMA, P. JAYASHREE et S. JIWANI : A survey on clustering current status and challenging issues. *International Journal on Computer Science and Engineering*, 2(9):2976–2980.

[97] S. RAMASWAMY, R. RASTOGI et K. SHIM : Efficient algorithms for mining outliers from large data sets. *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, p. 427–438, 2000.

[98] D. A. REYNOLDS, T. F. QUATIERI et R. B. DUNN : Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1-3):19–41, 2000.

[99] M. ROHAN : *Using Finite Mixtures to Robustify Statistical Models*. Thèse de doctorat, University of Waikato, 2011.

[100] P. J. ROUSSEEUW et K. V. DRIESSEN : A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[101] P. J. ROUSSEEUW et M. HUBERT : Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 1(1):73–79, 2011.

[102] A. R. RUNNALLS : Kullback-Leibler approach to Gaussian mixture reduction. *IEEE Transactions of Aerospace and Electronic Systems*, 43(3):989–999, 2007.

[103] N. F. SAMATOVA, G. OSTROUCHOV, A. GEIST et A. V. MELECHKO : Rachet : An efficient cover-based merging of clustering hierarchies from distributed datasets. *Distributed and Parallel Databases*, 11(2):157–180, 2002.

[104] O. M. SAN, V. HUYNH et Y. NAKAMORI : An alternative extension of the k-means algorithm for clustering categorical data. *International Journal of Applied Mathematics and Computer Science*, 14(2):241–247, 2004.

[105] J. SANDER, M. ESTER, H. P. KRIEGEL et X. XU : Density-based clustering in spatial databases : The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[106] G. SCHWARZ : Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[107] J. SEN : A robust and secure aggregation protocol for wireless sensor networks. *In Proceedings of the Sixth IEEE International Symposium on Electronic Design*, p. 222–227, 2011.

[108] A. STREHL et J. GHOSH : Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3(3):583–617, 2002.

[109] S. SUNDARAM et C. N. HADJICOSTIS : Distributed function calculation via linear iterations in the presence of malicious agents - part ii : Overcoming malicious behavior. *In In American Control Conference*, p. 1357–1362, 2008.

[110] J. TANG, Z. CHEN, A. W. C. FU et D. W. L. CHEUNG : Enhancing effectiveness of outlier detections for low density patterns. *In Advances in Knowledge Discovery and Data Mining*, p. 535–548, 2002.

[111] N. UEDA, R. NAKANO, Z. GHAHRAMANI et G. E. HINTON : Split and merge em algorithm for improving gaussian mixture density estimates. *VLSI Signal Processing*, 26(1):133–140, 2000.

[112] J. VAIDYA et C. CLIFTON : Privacy-preserving -means clustering over vertically partitioned data. *In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 206–215, 2003.

[113] M. Van der LAAN, K. POLLARD et J. BRYAN : A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation*, 73(8):575–584, 2003.

[114] N. VASCONCELOS : Image indexing with mixture hierarchies. *In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, p. 3–10, 2001.

[115] N. VLASSIS, Y. SFAKIANAKIS et W. KOWALCZYK : Gossip-based greedy gaussian mixture learning. *In Proceedings of the 10th Panhellenic conference on Advances in Informatics*, p. 349–359, 2005.

[116] S. VOULGARIS, D. GAVIDIA et M. van STEEN : Cyclon : Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2):197–217, 2005.

[117] S. VOULGARIS, M. JELASITY et M. van STEEN : A robust and scalable peer-to-peer gossiping protocol. *In In 2nd International Workshop Agents and Peer-toPeer Computing*, p. 47–58, 2003.

[118] S. VOULGARIS, M. van STEEN et K. IWANICKI : Proactive gossip-based management of semantic overlay networks. *Concurrency and Computation : Practice and Experience*, 19(17):2299–2311, 2007.

[119] J. WOLFE, A. HAGHIGHI et D. KLEIN : Fully distributed em for very large datasets. *In Machine Learning, Proceedings of the Twenty-Fifth International Conference*, p. 1184–1191, 2008.

[120] L. XIAO, S. P. BOYD et S. J. KIM : Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, 2007.

[121] R. XU et D. WUNSCH : Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[122] X. XU, M. ESTER, H. P. KRIEGEL et J. SANDER : A distribution-based clustering algorithm for mining in large spatial databases. *In Proceedings of the Fourteenth International Conference on Data Engineering*, p. 324–331, 1998.

[123] X. XU, J. JÄGER et H. P. KRIEGEL : A fast parallel clustering algorithm for large spatial databases. *Data Mining Knowledge Discovery*, 3(3):263–290, 1999.

[124] W. X. Y. WENG et L. XIE : Diffusion-based em algorithm for distributed estimation of gaussian mixtures in wireless sensor networks. *Sensors*, 11(6):6297–6316, 2011.

[125] B. ZHANG, M. HSU et G. FORMAN : Accurate recasting of parameter estimation algorithms using sufficient statistics for efficient parallel speed-up. *In in 4th European Conference on Principles and Practices of Knowledge Discovery in Databases*, p. 243–254, 2000.

[126] K. ZHANG, S. SHI, H. GAO et J. LI : Unsupervised outlier detection in sensor networks using aggregation tree. *In Advanced Data Mining and Applications, Third International Conference*, p. 158–169, 2007.

[127] T. ZHANG, R. RAMAKRISHNAN et M. LIVNY : Birch : An efficient data clustering method for very large databases. *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, p. 103–114, 1996.

[128] Y. ZHANG, N. MERATNIA et P. HAVINGA : Outlier detection techniques for wireless sensor networks : A survey. *Communications Surveys and Tutorials, IEEE*, 12(2):159–170, 2010.

[129] Y. ZHANG, N. MERATNIA et P. J. M. HAVINGA : A taxonomy framework for unsupervised outlier detection techniques for multi-type data sets. Rap. tech., Centre for Telematics and Information Technology University of Twente, 2007.

[130] Z. ZHANG, C. CHEN, J. SUN et K. L. CHAN : Em algorithms for gaussian mixtures with split-and-merge operation. *Pattern Recognition*, 36(9):1973–1983, 2003.

[131] J. ZHOU, C. ZHAO, Y. WAN, W. HUANG, B. YANG et J. GE : A novel outlier detection algorithm for distributed databases. *In Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, p. 293–297, 2008.

Liste des publications

Revues internationales

A. EL ATTAR, A. PIGEAU et M. GELGON : **A decentralized and robust approach to estimating a probabilistic mixture model for structuring distributed data**. (soumis à la revue internationale Web Intelligence and Agent Systems, sur invitation, version étendue d'une sélection d'articles du congrès IEEE/WIC/ACM/, Lyon, France, Août 22-27, 2011).

Conférences internationales

- A. EL ATTAR, A. PIGEAU et M. GELGON : **Fast aggregation of student mixture models**. In European Signal Processing Conference (Eusipco 2009), p.312Ű216, Glasgow, Scotland, Août 24-28, 2009. (orale)

- A. EL ATTAR, A. PIGEAU et M. GELGON : **A decentralized and robust approach to estimating a probabilistic mixture model for structuring distributed data**. In Proceedings of the 2011 IEEE/WIC/ACM International Conference on Web Intelligence, p. 372Ű379, Lyon, France, Août 22-27, 2011. IEEE Computer Society 2011. (orale)

- A. EL ATTAR, A. PIGEAU et M. GELGON : **A decentralized technique for robust probabilistic mixture modelling of a distributed data set**. In Intelligent Distributed Computing V - Proceedings of the 5th International Symposium on Intelligent Distributed Computing, p. 271Ű276, Delft, Netherlands, Octobre 5-7, 2011. Studies in Computational Intelligence 382 Springer 2012. (orale)

Liste des figures

Table des matières

Estimation robuste des modèles de mélange sur des données distribuées

Ali EL ATTAR

Résumé

Cette thèse propose une contribution en matière d'analyse de données, dans la perspective de systèmes informatiques distribués non-centralisés, pour le partage de données numériques. De tels systèmes se développent en particulier sur internet, possiblement à large échelle, mais aussi, par exemple, par des réseaux de capteurs. Notre objectif général est d'estimer la distribution de probabilité d'un jeu de données distribuées, à partir d'estimations locales de cette distribution, calculées sur des sous-jeux de données locaux. En d'autres termes, il s'est agi de proposer une technique pour agréger des estimés locaux pour en faire un estimé global. Notre proposition s'appuie sur la forme particulière que doivent prendre toutes les distributions de probabilité manipulées : elles doivent se formuler comme un mélange de lois gaussiennes multivariées. Notre contribution est une solution à la fois décentralisée et statistiquement robuste aux modèles locaux aberrants, pour mener à bien l'agrégation globale, à partir d'agrégations locales de mélanges de lois gaussiennes. Ces agrégations locales ne requièrent un accès qu'aux seuls paramètres des modèles de mélanges, et non aux données originales.

Mots-clés : clustering ; modèle de mélange, agrégation des modèles des mélanges ; estimation robuste ; détection de données atypiques ; données distribuées.

Abstract

This work proposes a contribution aiming at probabilistic model estimation, in the setting of distributed, decentralized, data-sharing computer systems. Such systems are developing over the internet, and also exist as sensor networks, for instance. Our general goal consists in estimating a probability distribution over a data set which is distributed into subsets located on the nodes of a distributed system. More precisely, we are at estimating the global distribution by aggregating local distributions, estimated on these local subsets. Our proposal exploits the following assumption: all distributions are modelled as a Gaussian mixture. Our contribution is a solution that is both decentralized and statistically robust to outlier local Gaussian mixture models. The proposed process only requires mixture parameters, rather than original data.

Keywords: clustering; mixture of models; aggregation of mixture models; robust estimation; outlier detection; distributed data.