Structuration géo-temporelle de données multimédia personnelles en vue de la navigation sur un appareil mobile

Antoine Pigeau

École Centrale de Nantes Université de Nantes École des Mines de Nantes

ÉCOLE DOCTORALE STIM
« SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DES MATÉRIAUX »

Année 2005

Structuration géo-temporelle de données multimédia personnelles en vue de la navigation sur un appareil mobile

---

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE NANTES

Discipline : Informatique

présentée et soutenue publiquement par

Antoine PIGEAU

le 9 décembre

à la Faculté des Sciences de l'Université de Nantes

devant le jury ci-dessous

Président	:	Michel CRUCIANU, PROFESSEUR	CNAM, Projet INRIA IMEDIA
Rapporteurs	:	Patrick GROS, CR CNRS	IRISA, Projet INRIA TEXMEX
		Hervé MARTIN, Professeur	Université Joseph Fourier, Grenoble
Examinateurs	:	Marc GELGON, Maître de Conférences	École Polytechnique de l'Université de Nantes
		Noureddine MOUADDIB, Professeur	École Polytechnique de l'Université de Nantes
		Andreas MYKA, Manager R&D	Nokia Group Finlande

Directeur de thèse : Noureddine MOUADDIB
Encadrants de thèse : Marc GELGON
Laboratoire : LINA (Laboratoire d'Informatique de Nantes Atlantique)

N° ED 366-233

STRUCTURATION GÉO-TEMPORELLE DE DONNÉES MULTIMÉDIA PERSONNELLES EN VUE DE LA NAVIGATION SUR UN APPAREIL MOBILE

Geo-temporal structuring of personal multimedia collection acquired and browsed from a mobile device

Antoine PIGEAU

favet neptunus eunti

Université de Nantes

Antoine PIGEAU

Structuration géo-temporelle de données multimédia personnelles en vue de la navigation sur un appareil mobile

x+178 p.

Ce document a été préparé avec LaTeX2$_\varepsilon$ et la classe `these-IRIN` version 0.92 de l'association de jeunes chercheurs en informatique LOGIN, Université de Nantes. La classe `these-IRIN` est disponible à l'adresse :

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

Impression : template-these.tex – 6/3/2005 – 16:19

Révision pour la classe : $Id: these-IRIN.cls,v 1.3 2000/11/19 18:30:42 fred Exp

Remerciements

Je tiens à remercier mes encadrants pour leur aide précieuse, les membres du jury pour avoir accepté de participer à ma soutenance de thèse et enfin ma famille et mes ami(e)s pour leurs encouragements.

Je tiens aussi à remercier toute l'équipe ATLAS-GRIM pour leur soutien et leur amitié.

Sommaire

Introduction

Cette section positionne nos travaux dans le domaine de la structuration de données personnelles. Nous y introduisons notre objectif et présentons le plan du document.

La gestion de données multimédia personnelles est un domaine de recherche très actif, qui prend racine dans un article de Vannevar Bush paru en 1945 [24]. Le problème est posé comme suit : pouvoir fournir "un appareil permettant à un individu de stocker ses livres, sa musique, ses communications, et qui puisse être consulté facilement". Autrefois trop peu évoluée, l'informatique a maintenant la puissance nécessaire pour envisager de tels appareils.

De nombreux projets sont consacrés à l'indexation de données personnelles [39, 46, 52, 64, 92]. En particulier, le projet Memex (Microsoft Research Digital Memories) propose de concevoir un système permettant de récupérer toutes les informations individuelles de notre vie. L'utilisateur est habillé de capteurs pour récupérer toutes sortes de données multimédia (audio, image, vidéo) et un logiciel spécifique permet d'enregistrer toutes sortes de données personnelles de son PC (ses mails, les pages webs consultées, ...). L'objectif est de concevoir un journal personnel de l'utilisateur à partir de toutes ses données. Les problèmes à résoudre sont :

- le développement des capteurs ;
- le stockage des données ;
- leur recherche et leur indexation ;
- l'analyse de contenu ;
- le développement de l'*Interaction Homme-Machine* (IHM) pour les utilisateurs ;
- les différentes applications ;
- la sécurité des données et leur confidentialité.

Dans cette thèse, nous nous focalisons sur l'organisation de collections d'images personnelles. Un journal personnel est susceptible de contenir un nombre important d'images : la génération de ce type de données est facile (contrairement à des données textuelles), les problèmes de stockage restent limités (contrairement à la vidéo) et leur contenu est explicite pour mémoriser les différents événements d'une vie. Parmi les axes de recherche précédent, les travaux de cette thèse ont porté sur le développement de nouveaux outils pour la recherche et l'indexation des images.

L'évolution technologique des appareils photographiques a ouvert de nouveaux horizons pour l'indexation des images dans le cadre des collections personnelles. La démocratisation des appareils photographiques numériques et leurs mises en oeuvre matérielles dans un nombre croissant d'appareils mobiles (téléphone, walkman, PDA (Personal Digital Assistant), ...) permettent maintenant à un utilisateur de disposer continuellement d'un capteur photographique numérique. Cette disponibilité, ajoutée à la

gratuité des images et aux applications multimédia émergentes (par exemple les messages multimédia) a augmenté la possibilité du nombre de prises de vue et entraîné un changement de comportement des utilisateurs face à l'image. En plus du nombre important d'images, le contenu en lui-même présente des différences avec des collections contenant des images obtenues à l'aide d'appareils argentiques. Auparavant essentiellement composée d'images souvenirs, une collection comprend maintenant de plus en plus d'images *anodines*. La collection d'images est ainsi similaire à un journal personnel, susceptible d'être alimentée et parcourue régulièrement du fait de sa disponibilité sur l'appareil mobile.

Une besoin est de proposer des outils pour retrouver efficacement des images dans une collection d'images personnelles.

Le domaine de l'indexation et la recherche d'images par le contenu est très étudié et présente de nombreux travaux [58, 104] surtout sur des collections *standards* (nous définissons une collection *standard* par une collection d'images ayant des sources diverses). La recherche d'une image par son contenu (par exemple la texture ou la couleur) est généralement motivée par un manque de métadonnées et nous allons voir que ces différentes techniques sont peu adaptées dans le cas de collections d'images personnelles. L'organisation de collections personnelles présente des particularités à prendre en compte. En particulier, les critères de recherche favoris des utilisateurs diffèrent de ceux proposés pour les collections *standards* (voir section 1.2, page 6). De nouvelles approches sont rendues possibles par de nouvelles métadonnées fournies par le support d'acquisition des appareils numériques. La date et le lieu de prises de vue, notamment, sont maintenant disponibles et d'après des études sur des utilisateurs [95] sont plus pertinents pour parcourir une collection d'images personnelles que la plupart des critères abstraits basés sur le contenu. Dans cette thèse, nous nous basons uniquement sur le lieu et la date de prises de vue pour organiser les images.

La recherche sur l'organisation de collections personnelles est en ce moment très active. De nombreux travaux industriels et académiques ont été récemment proposés, par exemple les systèmes Nokia Lifeblog [79] et Microsoft MyLifeBits [52]. Les différentes techniques proposées restent néanmoins très dépendantes de paramètres à régler manuellement, et aucune ne propose de solution pour une application dédiée aux terminaux mobiles. Notre objectif est ainsi de développer un système entièrement automatique pour organiser une collection d'images personnelles, en prenant en compte les contraintes liées aux terminaux mobiles, afin de pouvoir fournir un système approprié pour ce type d'appareil.

L'objectif général de cette thèse consiste à fournir des outils de classification spécifiques pour une collection d'images personnelles acquises et consultées d'un terminal mobile. Nous choisissons de formuler ce problème comme une tâche de classification automatique. Dans sa conception, nous nous efforçons de conserver un caractère non-supervisé. Beaucoup de systèmes existants traitant à peu près les mêmes données en vue du même objectif appliquent des paramètres qui nous paraissent arbitraires. Nous tâcherons, au contraire, de diriger au maximum la classification par les données.

Plan de la thèse

Le chapitre 1 présente les propriétés d'une collection d'images personnelles et un panorama des techniques, proposées dans la littérature pour organiser une telle collection. Le chapitre 2 établit un cahier des charges de notre tâche de classification, en vue de choisir un cadre méthodologique. Les principales approches pour la classification sont caractérisées, et le choix sur les modèles de mélange est motivé. Le chapitre 3 présente la modélisation de notre système et fournit les outils nécessaires pour manipuler et comprendre notre approche sur les modèles de mélange. Le chapitre 4 détaille les différents critères statistiques disponibles pour comparer la pertinence des partitions de modèles de mélange et valide notre choix sur le critère ICL (*Integrated Completed Likelihood*). Le chapitre 5 propose un algorithme in-

crémental d'optimisation du critère ICL pour construire des partitions des images. Cet algorithme est ensuite combiné dans le chapitre 6 avec un algorithme hiérarchique pour fournir notre propre algorithme hiérarchique incrémental. Les expériences validant notre approche sont détaillées dans le chapitre 7. Enfin, le chapitre 8 est consacré à la proposition de plusieurs techniques pour combiner les classifications obtenues en vue d'une IHM pour parcourir la collection sur un appareil mobile.

CHAPITRE 1

Propriétés et organisation automatique de collections d'images

Nous proposons dans ce chapitre un état de l'art sur l'indexation des collections d'images personnelles. Leurs propriétés sont mises en valeur afin de motiver les choix de notre approche, abordée dans le chapitre suivant.

1.1 Introduction

L'évolution des appareils photographiques et leurs implémentations sur des appareils mobiles ont entraîné un changement de comportement des utilisateurs de part les propriétés même du support d'acquisition (disponibilité du capteur photographique, *gratuité* de l'image, nouveaux services pour partager les images). Une collection numérique est ainsi susceptible de contenir plus d'images qu'une collection argentique et d'être parcourue plus régulièrement. Les capteurs numériques facilitent la prise de vue des images et la disponibilité de l'appareil mobile permet un accès permanent à sa collection. Il existe un besoin important de fournir des outils adaptés pour l'exploration d'une telle collection.

Les collections d'images personnelles présentent des propriétés différentes des collections *standards*. Ces différences sont dues aux critères favoris de recherche d'images, au processus de création de la collection et aux nouvelles métadonnées disponibles. Une recherche d'images dans une collection *standard* est généralement réalisée à l'aide d'une requête avec des critères basés sur le contenu des images (texture, couleur, ...). Dans notre contexte, les expériences de Rodden [95] sur des utilisateurs ont montré que les critères d'organisation favoris étaient la date ou le lieu des images, ce qui diffère sensiblement des critères précédents. L'utilisateur ayant une connaissance partielle de la collection, le parcours par lieu ou date de la collection est possible et même plus pertinent qu'une requête puisque l'utilisateur prendra plaisir à redécouvrir les différents événements vécus. Le support d'acquisition des images incluant automatiquement la date et bientôt le lieu des images, de nouvelles approches ont été possibles.

De nombreux travaux sur l'indexation des collections d'images personnelles sont maintenant proposés dans la littérature, industrielle comme académique. Néanmoins, comme nous le verrons, peu d'entre eux proposent un système entièrement automatique et ces systèmes dépendent généralement de paramètres gênants. De tels paramètres sont réglés manuellement et ont un large impact sur l'organisation obtenue. Les collections étant différentes suivant les utilisateurs, ces systèmes présentent de plus ou

moins bons résultats selon leur réglage. L'objectif est ainsi de construire un modèle pour lequel une paramétrisation *universelle* existe. Nous choisissons de traiter ce problème comme une tâche de classification automatique.

Ce chapitre présente dans un premier temps un état de l'art sur les collections d'images personnelles et les différentes techniques académiques et industrielles existantes.

1.2 Propriétés des collections d'images personnelles acquises sur un appareil mobile

Cette section présente les particularités des collections d'images personnelles. Nous montrons que l'organisation de telles collections est un domaine d'étude à part entière. Nous détaillons tout d'abord le changement de comportement des utilisateurs lié aux appareils numériques. Ensuite, nous présentons les critères favoris d'organisation d'une collection personnelle qui ressortent d'études réalisées sur des utilisateurs. Enfin, nous abordons les nouvelles métadonnées disponibles sur les supports d'acquisitions.

1.2.1 Un changement de comportement

Les appareils numériques présentent de nombreux avantages pratiques sur les appareils argentiques, entraînant de nouveaux comportements face à la prise de vue d'images. Ce changement est dû à plusieurs propriétés de cette nouvelle technologie :

- la *gratuité* des images ;
- la taille des mémoires de stockage ;
- la disponibilité ;
- les nouveaux services liés aux données multimédia.

Nous explicitons quels sont les impacts de ces propriétés sur le comportement de l'utilisateur.

Facilité de la prise de vue

Les deux premiers points facilitent la création d'une collection d'images. Les appareils photographiques numériques sont équipés d'écran permettant de visionner les images obtenues, entraînant une certaine gratuité de la prise de vue : une image n'a plus besoin d'être imprimée pour être partagée et les images ratées peuvent être directement effacées. La taille des mémoires de stockage disponibles étant en constante croissance (de l'ordre du giga-octet aujourd'hui), le nombre de prises de vue durant un événement n'est en pratique plus limité. Les cartes mémoires peuvent maintenant contenir plusieurs centaines d'images en haute résolution. Contrairement à la photographie argentique, où la pellicule était une contrainte sur le nombre de prises de vue, l'utilisateur n'est plus freiné ni par le coût d'une image ni par un problème de stockage. Il est donc enclin à prendre un nombre plus important d'images. Notons que cela peut avoir des effets négatifs : il n'est pas rare maintenant de prendre plusieurs fois la même image afin d'être sûr qu'elle soit réussie. Ces avantages peuvent donc se retourner contre l'utilisateur au moment du tri de la collection.

Impact sur le contenu des images

La compacité des capteurs photographiques numériques a permis leur intégration dans un nombre croissant d'appareils mobiles. Les téléphones portables sont maintenant pratiquement tous équipés d'appareils photographiques, et les baladeurs et les PDA en seront bientôt aussi tous dotés. Cette intégration sur des objets que l'on emmène toujours avec soi, permet aux utilisateurs de disposer continuellement d'un appareil photographique à portée de la main. Cette disponibilité facilite la création d'images et entraîne aussi un changement sur les sujets de prises de vues. Une étude réalisée par IPSE marketing en décembre 2002 a montré que les personnes équipées de téléphones portables avec un appareil photographique l'utilisaient dans 80% des cas. Le contenu porte dans 42% des cas sur des images d'événements anodins pour des personnes extérieures (mais intéressantes pour l'utilisateur). D'autres sujets pertinents sont les membres de la famille, les ami(e)s ou soi-même, les animaux de compagnie et enfin les souvenirs de voyages. Cette dernière utilisation semble peu représentée mais peut être expliquée par la faible qualité des images obtenues pour le moment avec de tels appareils. Néanmoins les progrès techniques sont en cours puisque des appareils de plus de 1 million de pixels sont maintenant disponibles.

Les utilisateurs ont tendance à prendre plus d'images personnelles de leur vie de tous les jours. Une telle collection d'images devient similaire à un journal *intime* et est donc susceptible d'être régulièrement parcourue. L'intimité de la collection est renforcée avec l'indépendance de l'utilisateur pour obtenir et stocker les images. Il ne dépend plus d'une tierce personne pour tirer les images sur papier par exemple. L'accès aux images peut être facilement limité à l'utilisateur.

Le partage des images

Enfin, les nouveaux services émergents, tels que les services de Message Multimedia (MMS) qui permettent d'envoyer/recevoir des images avec son téléphone portable, ou encore les blogs sur Internet, favorisent la prise d'images. Partager ses images via un téléphone portable se fait maintenant très facilement grâce aux évolutions des réseaux terrestres ou satellites, l'augmentation du débit permettant un partage rapide des images. La figure 1.1 ci-après présente les différents moyens disponibles actuellement pour partager ses images à partir d'un appareil mobile. Un utilisateur peut les partager avec un autre appareil mobile ou directement sur Internet sur son site web, ou encore en les envoyant par mail. Le MMS est similaire aux Short Message Service (SMS) mais permet d'inclure des images dans le message, nous pouvons donc nous attendre à ce que le nombre d'images prises sur les téléphones portables explose si son succès est identique.

Il est ainsi pertinent de penser qu'un utilisateur se retrouvera dans un futur proche avec un nombre de plus en plus élevé d'images numériques, constituant une sorte de journal personnel susceptible d'être consulté régulièrement. Des outils spécifiques pour organiser et parcourir ce journal doivent donc être fournis pour faciliter cette tâche.

1.2.2 Les critères d'organisation et de recherche

Dans les collections standards, les critères classiques sont essentiellement basés sur le contenu des images. Des critères tels que la couleur, la texture ou la mise en correspondance de différentes prises de vue de la même scène [77] sont utilisés pour faire des requêtes ou classer les images. Des travaux [35, 95, 119, 122] ont montré que ces critères sont peu pertinents pour des collections d'images personnelles. Les expériences sur des utilisateurs [95] ont montré que les critères favoris pour chercher ou organiser les images étaient :

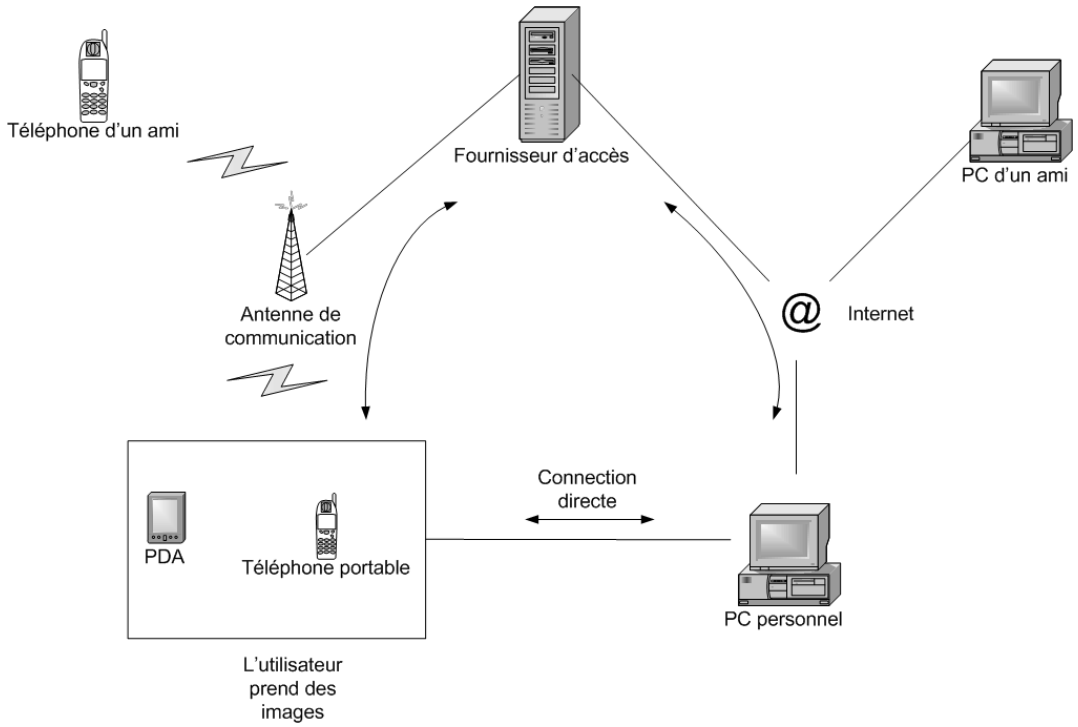


Figure 1.1: Partage d'images à partir d'un appareil mobile : l'utilisateur peut partager ses images en les envoyant sur d'autres appareils mobiles (MMS) ou en les rendant disponibles sur un site web (blog).

- la date ;
- le lieu ;
- la recherche par mots clés sur les annotations d'images.

Nous remarquons que ces critères sont tous liés à l'événement de la prise de vue et non au contenu même de l'image. Le tableau 1.1 ci-après résume les résultats obtenus dans les expériences de [95] : chaque critère est noté entre 1 et 5 (le score 1 est le plus élevé). Les critères basés sur le contenu des images ont des scores compris entre 2.8 et 3.2 alors que les critères favoris oscillent entre 1.5 et 1.7. Nous notons que le critère le plus apprécié est la recherche par mots clés sur les annotations des images. Le problème est que l'utilisateur ne prend généralement pas le temps d'annoter toutes ses images. Une telle approche devient peu efficace quand peu d'annotations sont disponibles.

Les utilisateurs organisent généralement leur collection par événement [32]. Un événement est généralement associé à un intervalle temporel ou un lieu. Une collection n'est donc pas une succession d'images isolées mais une succession de groupes d'images définis chacun par un événement précis. Cette notion est importante puisque pour rechercher une image, les utilisateurs recherchent en réalité l'événement durant lequel elle a été prise. Un groupe est associé à un intervalle de temps spécifique ou à un lieu, par exemple un anniversaire ou des vacances à l'étranger. La similarité entre deux images se base donc sur leur appartenance à un même événement, et non sur des similarités de contenu. Nous pensons néanmoins que les critères sur le contenu des images sont encore nécessaires pour organiser une collec-

Critères de recherche	note
lieu et date	1.7
annotations des images	1.5
couleur	3.2
texture	3.0
composition	2.8
recherche d'une image à partir d'une image similaire	2.3

Table 1.1: Résultat des expériences obtenu dans [95], sur la pertinence des critères de recherche d'images. Les critères sont notés entre 1 *et* 5*. Les critères favoris sont les annotations des images, le lieu et la date.*

tion d'images, mais seulement au sein même d'une classe. Par exemple, si un utilisateur prend beaucoup d'images durant une fête, il serait intéressant de classer ces images dans un événement précis puis de les classer selon leur contenu à l'intérieur même de ce groupe. Des travaux proposant le regroupement des images par similarité de leur contenu sont disponibles dans la littérature, par exemple [99], et restent pertinents pour notre application.

Un point important est la connaissance partielle de l'utilisateur sur l'ordonnancement des événements de la collection. Il crée lui-même sa collection d'images : le créateur et la personne la consultant ensuite sont en fait la même personne. Cette propriété est essentielle puisqu'elle influe sur la manière de rechercher une image. La recherche peut se faire en parcourant la collection plutôt qu'à l'aide d'une requête : une organisation par événements permet à l'utilisateur de sauter d'un événement à un autre jusqu'à celui comprenant l'image voulue. De plus, cette recherche offre le plaisir de redécouvrir différents souvenirs de sa vie.

L'approche par le contenu des images était essentiellement due au manque de métadonnées disponibles. Dans notre contexte, nous allons avoir accès à de nouvelles informations intéressantes pour organiser une collection personnelle.

1.2.3 Les métadonnées disponibles à partir d'un appareil mobile

Les appareils mobiles sont des ordinateurs possédant maintenant un système d'exploitation et une mémoire de stockage importante, et gèrent un nombre croissant de données multimédia. Ainsi pour chaque nouvelle image obtenue, des post-traitements sont réalisés sur le contenu des images (accentuation des couleurs, ...) et des métadonnées sont ajoutées automatiquement. Les constructeurs d'appareils photographiques numériques suivent des standards pour stocker ces informations. Dans le contexte des images numériques, le format EXIF (Exchangeable Image File) permet de mémoriser :

- la date de la prise de vue ;
- les informations liées à la prise de vue (temps d'exposition, ouverture du diaphragme, ...) ;
- les informations sur l'appareil (marque, ...).

On obtient ainsi facilement la date de l'image qui correspond à un critère pertinent pour l'utilisateur. Les informations liées à la prise de vue peuvent aussi être utilisées, comme nous le verrons dans la section

suivante (section 1.3).

Importance de la géolocalisation

Dans le contexte des téléphones portables et bientôt des appareils photographiques numériques, nous allons bientôt avoir accès à une propriété intéressante pour notre objectif qui est la géolocalisation [38, 106]. En effet, localiser un téléphone portable est déjà techniquement réalisable mais pour le moment réservé à des services spécialisés (service d'urgence par exemple). Les applications commerciales liées à la géolocalisation étant multiples et représentant un enjeu important, cette information sera prochainement accessible au grand public grâce par exemple au système de navigation satellite européen GALILEO. Un tel système a été développé pour fournir des informations concernant la position spatiale des utilisateurs dans différents secteurs tels que le transport (véhicules de location, contrôle de vitesse, navigation), les services sociaux (gestion des ambulances) ou les travaux publiques (informations géographiques). La géolocalisation sur des appareils mobiles est un point important et de nombreuses recherches le prouvent. Les systèmes [5, 41, 72, 73] permettent l'apprentissage et l'identification des lieux courants d'utilisateurs. [74, 103] proposent des solutions pour les visites touristiques ou des applications spécifiques sur la localisation sur des cartes géographiques. La géolocalisation peut aussi être utilisée pour vérifier les possibilités de communication entre personnes [73, 109]. Ces travaux motivent la disponibilité future de la géolocalisation dans des applications de la vie quotidienne.

Systèmes de géolocalisation

Les systèmes de géolocalisation permettent d'obtenir la position d'un appareil mobile avec beaucoup de précision (précision à 1 mètre). L'organisation de développement de standard 3G [106, 126] a développé des systèmes de localisation hybrides combinant les réseaux terrestres et le Global Positioning System [87], permettant d'améliorer la précision et la disponibilité de la géolocalisation. Un tel système peut être performant dans des environnements urbains (obstruction des satellites GPS, due aux immeubles par exemple, palliée par le réseau terrestre) comme en pleine nature (absence de réseaux terrestres palliée par le GPS).

Un exemple est la combinaison des techniques E-OTD (*Enhanced Observed Time Difference*) et A-GPS (*Assisted Global Positioning System*)[106]. Ces deux systèmes sont utilisés pour les réseaux GSM (*Global System for Mobile*) et GPRS (*General Packet Radio Service*). Le système E-OTD permet de déterminer la position d'une station mobile à partir de stations de base fixe, à l'aide d'une méthode de triangularisation. La station mobile doit mesurer les différences de temps d'accès à au moins trois stations de base afin de pouvoir récupérer sa position. Le système A-GPS fonctionne sur le même principe mais à partir de satellites. Ce système permet de faciliter la géolocalisation GPS en faisant appel aux possibilités de communication du terminal mobile [76].

Notons que dans le contexte des appareils photographiques, l'intégration de systèmes GPS est imminente : l'entreprise Ricoh est sur le point de sortir un appareil intégrant un système GPS. Les images obtenues répondraient à la norme GIS (Graphical Information System), qui permet de visualiser, manipuler, analyser et afficher des données géographiques.

Dans nos travaux nous ne nous intéressons pas aux détails techniques pour acquérir les coordonnées géographiques des images. Nous supposons que cette information est disponible facilement, avec une bonne précision.

Les nouvelles métadonnées accessibles sur un appareil mobile, la localisation et la date, correspondent aux critères de recherche pertinents pour les utilisateurs. Elles ont permis de nouvelles approches pour organiser les collections d'images personnelles.

Figure 1.2: Exemple d'interface du logiciel Nokia LifeBlog : le système permet d'organiser les données multimédia obtenues sur le téléphone mobile (SMS, image, vidéo). [79]

1.3 État de l'art sur les systèmes d'organisation de collections d'images personnelles

Nous assistons aujourd'hui à l'émergence de systèmes de gestion d'images personnelles. Des systèmes fonctionnels comme le *Microsoft World Wide Media eXchange* [114], *Hello* de Picasa (disponible sur www.hello.com) ou des prototypes tels que *The Mobile Media Meta-data* [35] et le système *LOCALE* [81], proposent de rechercher des images personnelles selon le contenu des images ou leurs métadonnées.

Des systèmes plus particuliers, liés aux téléphones portables équipés d'appareils photographiques, commencent aussi à se développer. Le projet *Nokia LifeBlog* [79] crée un journal personnel de l'utilisateur automatiquement en se basant sur ses images et ses SMS. La figure 1.2 ci-dessus présente une capture d'écran de ce système. Un autre exemple est le système *Cognima* (www.cognima.com) qui donne la possibilité de partager ses images sur le web. L'existence de ces nouvelles applications confirme un besoin des utilisateurs pour la gestion de leurs images numériques. Néanmoins, les techniques d'organisation proposées par ces systèmes restent simplistes puisque aucune ne propose de classification automatique.

Des universités et des industriels ont proposé des solutions pseudo-automatiques pour organiser des collections d'images personnelles. Plusieurs approches ont été proposées, allant de l'organisation manuelle basée seulement sur les métadonnées jusqu'à des techniques entièrement automatiques. Nous détaillons tout d'abord les approches basées sur les métadonnées, ensuite sur le contenu, et enfin les organisations temporelles et spatiales.

1.3.1 Organisation d'une collection d'images à l'aide de métadonnées

Certains systèmes utilisent seulement les métadonnées pour organiser les images. L'objectif étant de faciliter/automatiser l'annotation des images et de pouvoir rechercher une image donnée à l'aide d'une requête.

Des techniques simples ont été proposées pour faciliter la création de métadonnées. Le système [101] permet de nommer les personnes présentes dans des images en faisant glisser leur nom avec la souris et [6] propose d'enregistrer des commentaires audios sur les images. Néanmoins, ces manipulations demandent du temps et sont à réaliser pour chaque image. Il est donc peu probable que l'utilisateur les réalisent pour chaque prise de vue.

Afin de générer automatiquement les métadonnées, une première approche consiste à récupérer les informations sur le contexte de l'image [82]. Le lieu et la date permettent de retrouver plusieurs informations contextuelles :

- le pays, la région, la ville, . . . ;
- le climat (nuageux, température,. . .) ;
- la saison, le mois, le jour, ...

La figure 1.3 ci-après présente un exemple d'interface du système [82]. Les informations météorologiques peuvent être récupérées sur Internet ou bien à partir d'un SIG [36]. Tous les champs générés ne sont pas pertinents. En effet, un utilisateur ne se souvient pas forcément du climat qu'il faisait au moment de prendre sa photographie et encore moins de l'altitude.

Une seconde approche est de partager les métadonnées via un réseau d'utilisateurs [98, 122]. En se basant sur le contexte de l'image, une recherche automatique est alors réalisée parmi les annotations d'images ayant un contexte similaire. Ces annotations sont proposées automatiquement et doivent être validées par l'utilisateur. L'avantage sur l'approche précédente est l'accès à des métadonnées plus riches (un commentaire sur le contenu de l'image d'un autre utilisateur par exemple). Un tel système suppose néanmoins un grand nombre d'utilisateurs pour être efficace et requiert l'attention de l'utilisateur pour chaque annotation automatique.

Enfin, une annotation automatique de chaque image peut être réalisée en se basant sur le contenu de l'image [78]. Ces techniques manquent encore de précision, à cause de la difficulté à déterminer une annotation pertinente en se basant juste sur un critère abstrait comme la couleur ou la texture. Dire qu'une zone de ciel est présente car du bleu est situé sur le haut d'une image n'est pas toujours vrai.

Ces approches sont pertinentes pour enrichir les métadonnées des images et proposer des systèmes de recherche par requête. L'organisation par événements n'est en revanche pas proposée. Néanmoins, ces approches restent pertinentes pour pouvoir ensuite étiqueter un groupe d'images avec des métadonnées facilement interprétables. Dans nos travaux, nous proposons tout d'abord de classer les images et ensuite de représenter les groupes obtenues à l'aide de métadonnées. Un exemple d'une telle approche est aussi proposée dans [83] : elle consiste à annoter les personnes présentes sur les images à partir de classifications temporelle et spatiale.

1.3.2 Organisation d'une collection d'images par le contenu

Plusieurs systèmes proposent une approche basée sur le contenu des images. Des critères abstraits, tels que la couleur ou la texture, et la détection de visage sont proposés.

Comme nous l'avons vu précédemment, ces critères sont peu pertinents pour les utilisateurs. Ces systèmes combinent donc généralement cette approche avec d'autres critères. Les systèmes [69, 89, 90] organisent les images temporellement et affinent le résultat à l'aide du contenu des images. Le système [17] propose un critère de similarité combinant le contenu des images et leurs paramètres optiques (temps

Figure 1.3: Exemple de métadonnées récupérables avec une base de connaissances ou sur Internet, à partir de la date et du lieu [82].

de pose, diaphragme, ...). Ces paramètres sont censées nous donner des informations sur le contexte de l'image : intérieur ou extérieur, portrait, ...

Une technique plus intéressante pour les utilisateurs est la détection de visages, proposée par exemple dans [50, 101]. Ce système permet la détection de visages et leur apprentissage dans le but de faciliter l'annotation des images. Pour chaque visage détecté sur une nouvelle image, le système fait une proposition d'annotation à l'utilisateur. Ce critère est apprécié des utilisateurs et pertinent pour rechercher les images à l'aide d'une requête. La mise en oeuvre d'une telle technique sur un appareil mobile est maintenant possible avec l'apparition d'algorithmes peu coûteux [118]. Néanmoins, une telle approche n'est pas suffisante pour parcourir ou organiser une collection d'images.

1.3.3 Structuration temporelle d'une collection d'images

L'organisation d'une collection à partir de la métadonnée temporelle est proposée dans de nombreux systèmes. Cela s'explique par l'accessibilité à cette information, celle-ci étant directement incluse dans les métadonnées de l'image par l'appareil photographique. Elle présente de plus l'avantage d'être facilement interprétable.

L'objectif de ces méthodes est de déterminer l'intervalle de temps séparant les événements de la collection. Une telle durée varie néanmoins suivant la structure de la collection. Un utilisateur peut photographier plusieurs événements sur un court laps de temps et ensuite sur des périodes beaucoup plus longues. Elle est aussi subjective : selon l'utilisateur, deux semaines de vacances peuvent être représentées par un ou plusieurs événements.

Une première méthode pour déterminer les groupes d'images consiste à fixer manuellement une limite inter-événements [90]. Les expériences de ces travaux ont montré que cette limite présentait des résultats corrects si elle variait entre 6 et 24 heures. L'avantage de cette méthode est sa simplicité de

mise en oeuvre. Cependant, sa dépendance à un paramètre fixé manuellement peut donner des résultats variables suivant les collections. Certains algorithmes [55] l'utilisent néanmoins pour initialiser leurs partitions avant de procéder à des traitements plus complexes. Ceux-ci calculent automatiquement une limite inter-événements variable en se basant sur la moyenne des intervalles de temps séparant les images deux à deux [32, 48, 69, 89]. Par exemple, le système Phototoc [89] se base sur la moyenne des intervalles de temps dans une fenêtre donnée et [69] utilise l'algorithme k-means sur un histogramme des distances temporelles des images deux à deux pour fixer une limite inter-événements.

Plusieurs de ces techniques sont couplées avec des critères de classification basés sur le contenu de l'image [17, 69, 89, 90]. Certaines utilisent aussi les caractéristiques de prise de vue des images [17, 49] comme la distance du sujet, l'ouverture de l'objectif ou le temps d'exposition.

Enfin plusieurs travaux [32, 55, 78] proposent des classifications hiérarchiques, construites à partir des métadonnées (division générale par année, mois, jour) ou en se basant sur plusieurs valeurs de limites inter-événements. Cette approche est intéressante puisqu'elle fournit plusieurs points de vue sur un épisode temporel et elle répond ainsi aux problèmes de définition d'un événement.

1.3.4 Organisation géographique d'une collection d'images

Il n'existe, à notre connaissance, qu'un seul système d'organisation d'images [81] utilisant les coordonnées géographiques pour classer automatiquement les images. Cette métadonnée est, en pratique, encore peu disponible pour les appareils grand public, ce qui explique le peu de propositions d'organisation spatiale (nous utilisons indifféremment les mots *spatial* et *géographique*). Ce système est celui se rapprochant le plus de nos travaux. Contrairement à notre approche, il se base sur une série de règles de "bon sens", pour déterminer les limites inter-événements. Par exemple, une distance géographique entre deux images supérieure à une limite fixée prédit un changement d'événements (technique identique à celle de fixer manuellement une limite inter-événement). Le système dépend de paramètres arbitraires.

Le point particulièrement intéressant de cette approche est qu'elle combine les informations temporelles et spatiales pour obtenir deux classifications distinctes : une temporelle et une spatiale. Les images sont tout d'abord classées à partir de la métadonnée temporelle puis, une classification de ces événements est ensuite réalisée en se basant sur le lieu (les événements sont affectés à différents lieux). La classification temporelle est initialisée à partir d'une limite inter-événement fixée manuellement et une approche probabiliste automatique permet ensuite de classer ces séries temporelles par lieu. Une hiérarchie de lieux est ensuite construite en appelant de façon récursive l'algorithme de classification sur les lieux comprenant un grand nombre d'événements. Le niveau le plus général est ensuite obtenu en se basant simplement sur une division par pays (un utilisateur se rappelant généralement dans quel pays les images ont été prises). Cette approche est néanmoins basée sur plusieurs critères fixés manuellement. Notre objectif est de fournir un système ne dépendant pas de tels critères.

1.4 Conclusion

Les collections d'images personnelles deviendront de plus en plus courantes et comprendront un nombre élevé d'images grâce à la démocratisation des appareils numériques et à leur implantation sur divers types d'appareils usuels. Un problème émergeant est donc l'organisation de ces collections.

Nous avons tout d'abord vu que les propriétés des collections personnelles diffèrent des collections standards suivant plusieurs points :

- les critères de recherche favoris : le lieu et la date ;
- la connaissance partielle de la collection par l'utilisateur ;
- la disponibilité de nouvelles métadonnées, correspondant aux critères favoris des utilisateurs.

Ces différences motivent une approche spécifique pour organiser une collection d'images personnelles. Ces propriétés sont prises en compte pour la proposition de notre système.

Ensuite nous avons détaillé les différentes approches existantes dans la littérature pour organiser une collection d'images personnelles. Les systèmes proposés sont basés sur des approches dépendant de paramètres arbitraires à régler manuellement. Notre principal objectif est d'améliorer ce point en proposant une approche automatique, basée sur un modèle universel de collection d'images personnelles.

CHAPITRE 2

Cahier des charges et choix de l'approche pour la classification

Nous présentons dans ce chapitre nos besoins pour fournir une organisation adéquate et validons notre choix sur une technique de classification. Un aperçu des principales approches pour la classification est proposé.

2.1 Introduction

Notre approche consiste à extraire des épisodes spatio-temporels de la collection, en se basant exclusivement sur la date et le lieu de prise de vue de chaque image. Nous avons vu dans le chapitre précédent que ces deux métadonnées correspondent aux critères favoris des utilisateurs pour parcourir leur collection. Nous choisissons de traiter ce problème comme une tâche de classification automatique.

Notre premier objectif est de déterminer une méthode de classification adéquate. De nombreuses techniques existent, et chacune présente des propriétés spécifiques. Le choix de l'approche est un point important et doit être décidé en fonction des caractéristiques des données à classer et du résultat voulu en sortie. Plusieurs solutions sont bien sûr possibles et nous mettrons en valeur celles présentant les propriétés les plus adaptées pour notre application.

Nous présentons tout d'abord les grandes lignes de notre approche. Ensuite, après avoir présenté les différentes familles de classifications, nous argumentons pour l'approche par modèle probabiliste.

2.2 Notre cahier des charges

Nous présentons dans cette section notre approche pour organiser une collection d'images personnelles sur un appareil mobile. Dans un premier temps, nous abordons les contraintes liées à un tel appareil et les conséquences que cela entraîne sur notre proposition. Ensuite, nous présentons une vue générale de notre solution.

2.2.1 Les contraintes liées aux terminaux mobiles

Pour que notre système soit adapté sur un terminal mobile, nous devons prendre en compte plusieurs contraintes liées à ce type d'appareil :

- l'interaction homme-machine (IHM) est restreinte. Le clavier est peu adapté pour parcourir des images et un écran ne permet pas l'affichage de beaucoup d'images ;

- l'autonomie en énergie est limitée. L'affichage d'un grand nombre d'images simultanément doit être évité et l'algorithme de classification ne doit pas présenter une trop forte complexité en temps de calcul ;

- la mémoire de stockage disponible est pour le moment trop faible (1 ou 2 Giga) pour contenir une grande collection (de plus la taille des images est pour le moment en constante croissance).

Un besoin important, qui découle de ces contraintes, est de pouvoir résumer une collection d'images. L'utilisateur a une connaissance partielle de sa collection : il est donc capable, à partir d'un résumé de plusieurs événements, de déterminer les images associées. Un parcours par résumés pourrait permettre des économies d'énergies (limitation de l'affichage d'images) et de stockage. Prenons l'hypothèse que l'utilisateur a une connexion directe à une unité de stockage accessible via le réseau, seuls les résumés de la collection ont alors besoin d'être disponibles sur l'appareil mobile. Quand il cherche une image, il parcourt d'abord les résumés et ensuite il télécharge l'image à partir de l'unité de stockage.

La faible autonomie impose aussi un algorithme de classification peu coûteux. La collection étant réalisée au fur et à mesure dans le temps, un système incrémental semble pertinent. Reconstruire entièrement les partitions pour chaque nouvelle image paraît inadapté au vu du nombre important d'images qu'une collection peut contenir.

2.2.2 Aperçu de notre technique

Notre objectif est de permettre à un utilisateur de parcourir sa collection d'images par événement. Un événement est défini soit par un intervalle de temps, soit par un lieu. Il faut bien noter que notre technique automatique n'a aucune prétention de retrouver des événements. Mais il se trouve que souvent, les événements correspondent à des images proches, spatialement ou temporellement.

Nous nous basons uniquement sur les métadonnées spatiales et temporelles pour organiser la collection d'images. Cette extraction d'épisodes spatiaux et temporels est motivée par les raisons suivantes :

- les axes temporels et spatiaux employés pour structurer la collection sont simples et familiers aux utilisateurs (similaires à un calendrier et à une carte géographique), contrairement à des critères tels que la couleur ou la texture des images ;

- le parcours, par opposition aux requêtes, permet à l'utilisateur de naviguer dans sa collection personnelle sans avoir d'objectif précis (pour le plaisir ou pour avoir un résumé des événements récents) ;

- enfin, le parcours selon ces axes est adapté aux contraintes d'IHM. Le clavier numérique peut convenir pour parcourir la collection.

Nous proposons de construire deux classifications hiérarchiques distinctes, une temporelle et l'autre spatiale, que nous combinons ensuite pour fournir une partition hybride spatio-temporelle.

Le choix d'une structure hiérarchique est motivé par les contraintes liées aux appareils mobiles et

la définition d'un événement, qui pourra ainsi être représenté suivant plusieurs niveaux de granularités. En effet, la notion de lieu ou d'événement n'est pas clairement définie et dépend des utilisateurs. Une semaine de vacances peut être considérée comme un seul événement temporel et un seul lieu, ou une succession d'événements et de plusieurs lieux.

La construction de ces classifications doit être incrémentale afin de prendre en compte le processus de création de la collection d'images et les contraintes d'autonomie des appareils mobiles. La figure 2.1 ci-après présente un exemple de résultats que nous voulons obtenir à partir des métadonnées temporelles et spatiales. Les différents lieux de la collection doivent être retrouvés et présentés selon plusieurs niveaux de détail. Un résultat similaire doit être obtenu pour l'organisation temporelle, chaque événement devant être convenablement délimité.

Motivation de l'indépendance des classifications

Nous choisissons de garder une indépendance entre l'extraction des épisodes temporels et spatiaux. Les classes temporelles et spatiales étant liées, une classe spatiale peut être composée de plusieurs événements temporels, puisque les lieux usuels peuvent être visités plusieurs fois. Inversement, un événement temporel peut être composé de plusieurs lieux (vacances, mariage). La supposition qu'un changement de lieu entraîne un changement d'événements temporels est généralement vérifiée mais présente aussi des exceptions. Garder une indépendance entre les classifications spatiales et temporelles présente donc un avantage puisque cela permettra de détecter les événements temporels comprenant plusieurs lieux, et réciproquement.

Une alternative à cette approche est de ne construire qu'une seule classification à partir des métadonnées temporelles et spatiales (donc en 3 dimensions). Mais elle peut poser un problème pour la recherche des images par le lieu. En effet, on perdrait le lien entre les images prises sur un même lieu et à des dates différentes. Obtenir deux classifications distinctes est plus pertinent, l'objectif étant ensuite de permettre la combinaison des deux classifications pour la recherche d'images. Au moins une des deux métadonnées étant retenue par l'utilisateur pour une image recherchée [119], il pourra combiner les partitions temporelles et spatiales, en passant de l'une à l'autre, pour obtenir l'épisode recherché. Par exemple, l'utilisateur cherchera un lieu précis dans la classification spatiale puis pourra obtenir les différents événements temporels liés à ce lieu grâce à la classification temporelle. Garder l'indépendance entre les partitions permet ensuite plusieurs solutions pour les combiner. Nous proposons dans le chapitre 8 deux méthodes de construction de partitions hybrides.

Les métadonnées contextuelles

Nous n'utiliserons pas de métadonnées contextuelles (par exemple, la saison, le jour ou le mois de la prise de vue) pour **directement** classer les images. Nous proposons de construire tout d'abord des partitions à partir des métadonnées temporelles et spatiales et ensuite d'utiliser les métadonnées contextuelles. Une approche basée seulement sur ces dernières nous semble peu pertinente si elle n'est pas couplée avec une autre méthode. Les problèmes se posent généralement dans le cas où un événement se déroule aux limites des valeurs de métadonnées. Par exemple, si l'on construit un arbre de décision sur les métadonnées contextuelles, un événement ayant lieu sur deux jours sera divisé en deux événements distincts : un pour chaque jour. De plus, quand le nombre d'images est trop important, cette approche devient inefficace pour parcourir la collection. On peut par exemple obtenir un nombre important d'images avec les mêmes métadonnées pour le lieu. Dans le cas temporel, l'utilisateur ne se souvient pas toujours à quelle date il a pris l'image recherchée.

Nous proposons d'utiliser les métadonnées contextuelles après avoir obtenu nos classifications :

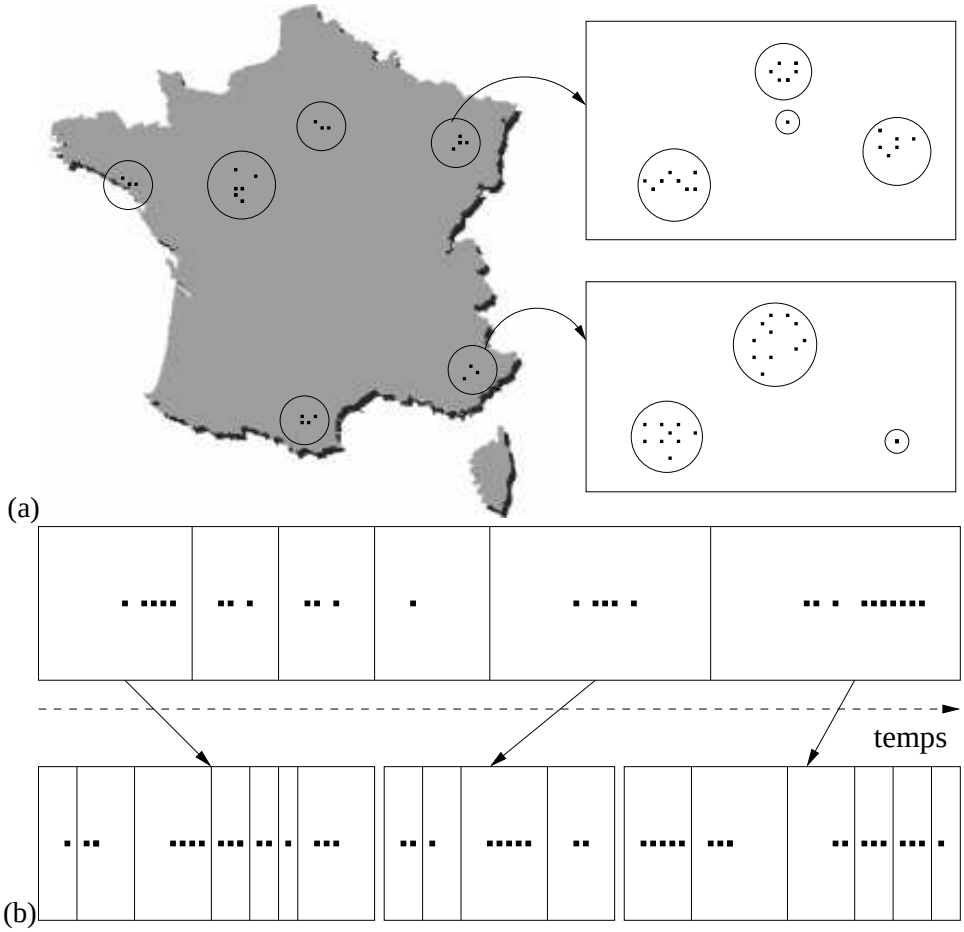


Figure 2.1: Exemple de classifications hiérarchiques spatiale et temporelle : les points représentent les métadonnées spatiales et temporelles. Les cercles sur (a) symbolisent les classes spatiales et les lignes sur (b), les limites entre les événements. Les flèches indiquent les liens de parenté entre les classes.

chaque classe est résumée à l'aide des métadonnées contextuelles des images qu'elle contient. L'intérêt est de :

- pouvoir représenter un événement à l'aide de labels textuels : ce point est intéressant dans le contexte des appareils mobiles du point de vue des contraintes d'IHM (taille de l'écran) et d'autonomie en énergie (l'affichage de texte est moins coûteux qu'une image) ;

- permettre à l'utilisateur de faire des requêtes directement sur les **événements** de sa collection (et non les images) : un utilisateur associant une image à un événement, une requête sur une image n'est pas pertinente et peut de plus présenter des résultats de moins bonne qualité (la représentation d'un événement à partir de son ensemble d'images associées est plus riche que pour une image seule et augmente ainsi les chances de bonnes réponses) ;

Les informations contextuelles à ajouter dans chaque image doivent être définies en fonction des besoins

des utilisateurs. Une base de connaissances et un *Système d'Information Géographique* (SIG) doivent être proposés par des experts. Un SIG est un système informatique permettant, à partir de diverses sources, de rassembler et d'organiser des informations localisées géographiquement.

Vue générale du processus d'organisation de la collection

La figure 2.2 ci-après présente les différentes étapes pour organiser la collection. Les premières étapes consistent à récupérer les métadonnées des images (lieu, date et informations contextuelles), à mettre les partitions à jour et enfin à proposer une partition hybride adéquate pour explorer la collection, prenant en compte les contraintes d'IHM. Pour chaque nouvelle image, les étapes de mise à jour sont :

1. récupération des coordonnées géographiques de l'image. Nous supposons que la localisation des images est fournie par un système du type A-GPS/E-OTD [106] ;

2. ajout des métadonnées contextuelles dans les métadonnées de l'image à partir d'une base de connaissances (pour la date) et d'un Système d'Information Géographique (SIG) ;

3. mise à jour des classifications spatiales et temporelles (seulement à partir de la date et du lieu de l'image) ;

4. construction d'une partition hybride spatio-temporelle.

Le parcours d'une structure hiérarchique étant complexe sur un appareil mobile, nous proposons de combiner les deux classifications (temporelle et spatiale) en une seule partition hybride spatio-temporelle afin de simplifier l'IHM : l'objectif est de permettre de parcourir simultanément les partitions temporelle et spatiale initiales.

Notre premier objectif est de développer un algorithme de classification hiérarchique et incrémental. Avant de présenter notre choix sur l'approche utilisée pour construire nos partitions, nous rappelons les différentes techniques de classification dans leur généralité.

2.3 Aperçu des principales approches pour la classification

Nous proposons une description générale des différentes techniques de classification existantes, présentées dans des états de l'art sur la question [8, 57, 62]. La figure 2.3 page 23 présente une taxonomie des algorithmes de classification existants. D'autres solutions sont évidemment possibles [63, 8], beaucoup d'algorithmes mélangeant les différentes approches. Nous nous bornerons à expliquer les algorithmes les plus classiques vis à vis des besoins que nous venons d'exprimer. Notre objectif est de mettre en valeur les différentes propriétés de chaque approche et de pouvoir ainsi motiver notre choix. Nous présentons rapidement chacun des algorithmes en se basant sur la figure 2.3. Nous détaillons les techniques les plus courantes et nous mettons en valeur leurs propriétés. Les points importants à mettre en valeur sur les algorithmes de classification sont :

- le type d'attribut géré ;

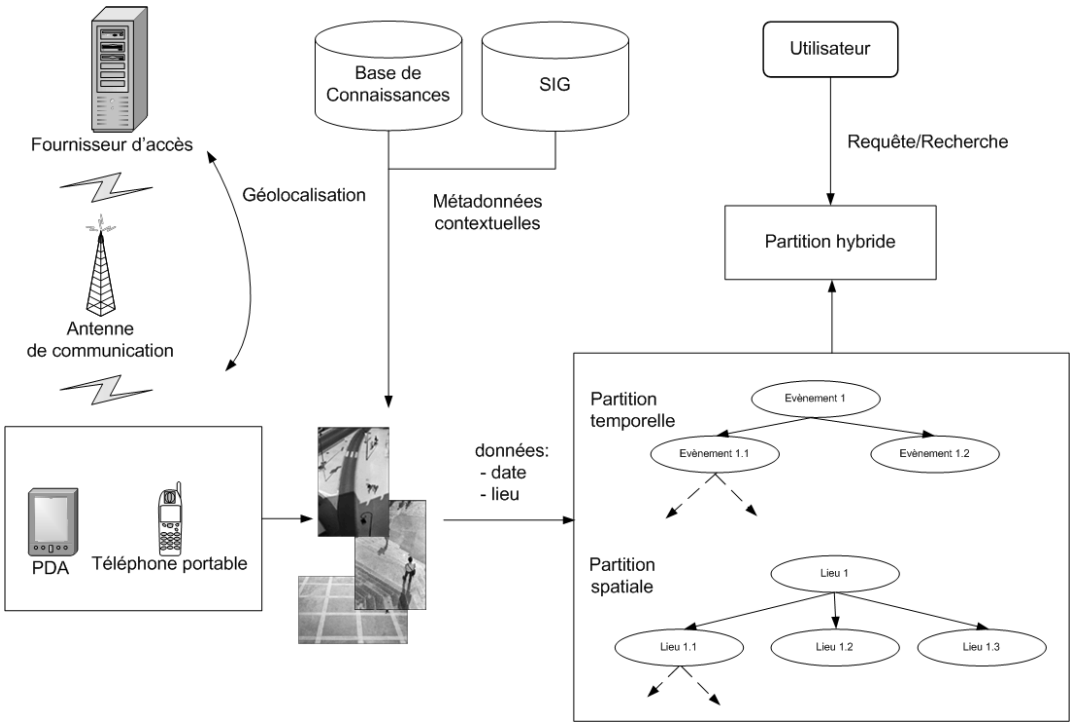


Figure 2.2: Organisation d'une collection d'images personnelle acquise à partir d'un appareil mobile : pour chaque nouvelle image, les métadonnées spatiales, temporelles et contextuelles sont ajoutées automatiquement. Nous supposons que la localisation des images est fournie par un système du type GPS/E-OTD. Les classifications temporelles et spatiales sont ensuite mises à jour incrémentalement et une partition hybride est proposée à l'utilisateur pour explorer la collection.

- l'habilité à fonctionner avec des données à grande dimension ;
- pour la classification spatiale, l'habilité à trouver des composantes de forme irrégulière ;
- la gestion des petits échantillons ;
- la complexité en temps de calcul ;
- l'affectation des données (statique ou dynamique) ;
- l'interprétation des résultats.

2.3.1 Algorithmes hiérarchiques

Les algorithmes hiérarchiques construisent un arbre de classes, appelé dendogramme. Chaque noeud a des composantes fils, leur ensemble représentant une partition plus fine que leur père. Deux méthodes différentes existent pour construire la hiérarchie : par agglomération (de bas en haut) ou par division

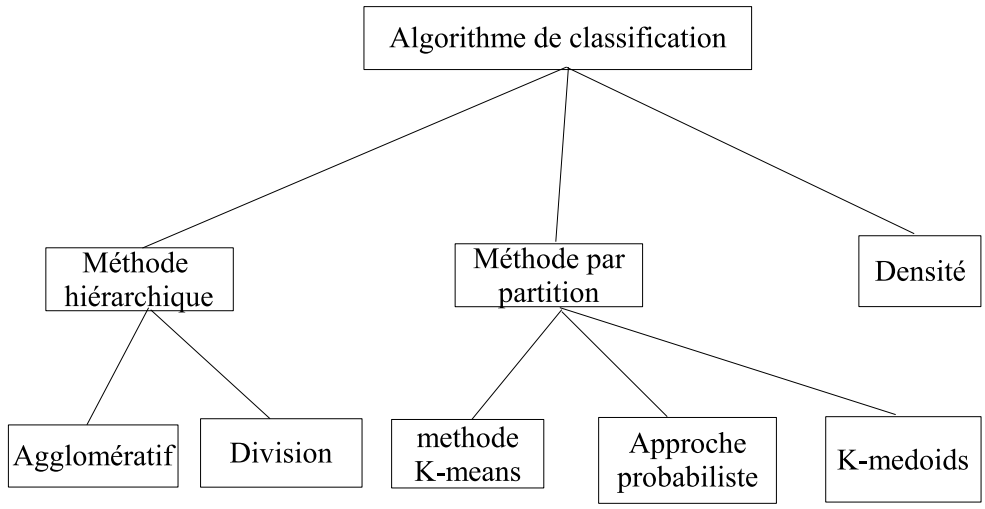


Figure 2.3: Proposition d'une taxinomie des algorithmes de classification inspirée de [63].

(de haut en bas). L'approche agglomérative est généralement initialisée avec une donnée par classe et fusionne deux classes à chaque itération. L'approche par division est initialisée avec une seule classe et itère des divisions successives. Le processus s'arrête quand un critère d'arrêt est vérifié (par exemple le nombre de classes). La figure 2.4 page 25 présente un exemple de dendogramme.

Les décisions de fusion ou division de composantes sont prises en se basant sur la similarité des composantes. Celle-ci est basée sur une matrice de distance des classes deux à deux, appelée *matrice de connectivité*. Les classes à fusionner ou à diviser sont choisies à partir de cette matrice. Son stockage est parfois coûteux quand le nombre de données à classer est important. Ce point ne nous concerne pas puisque le nombre d'images dans une collection personnelle reste limité par rapport aux collections *standards*.

Il est ensuite nécessaire de définir une mesure de similarité. Le type de mesure utilisé joue un rôle important sur la hiérarchie puisqu'il reflète les notions de connectivité et de proximité entre les classes. Ces algorithmes fonctionnent avec n'importe quelle fonction de distance. Ils peuvent donc être appliqués sur des données ayant des attributs de tous types.

La similarité entre deux sous-ensembles est basée sur la distance des données deux à deux, avec une donnée dans le premier ensemble et l'autre dans le second. La similarité entre deux ensembles $C1$ et $C2$ ($C1 \cap C2 = \emptyset$) est donc définie par :

$$d(C_1, C_2) = \textit{opération}\{d(x, y) | x \in C_1, y \in C_2\}, \tag{2.1}$$

où *opération* définit le type de similarité choisi et $d(x, y)$ est la distance entre x et y. Les distances les plus communes sont le *lien simple*, le *lien complet* et le *lien moyen*.

La distance avec le *lien simple* est définie par la distance minimum entre deux classes (opération$=$ min). Le *lien complet* est l'inverse, l'opération étant définie par le maximum des couples $d(x, y)$. Ces deux approches sont liées à la théorie des graphes. La matrice de connectivité peut être associée à un graphe $G = (X, E)$ complet où les sommets X sont les données et E les poids des arêtes, déterminés par les entrées de la matrice. Le *lien simple* revient à déterminer le graphe de poids minimum (*Maximum Spanning Tree*) et le *lien complet* se base sur le concept des cliques.

Enfin le *lien moyen* est plus complexe, une classe étant définie par une moyenne de ses données. L'opération peut être définie par la moyenne (le centre), la médiane ou la variance des données. Cette dernière méthode est plus appropriée aux données numériques et présente des ressemblances avec les algorithmes par partition, détaillés dans la section suivante.

Les inconvénients liés à l'approche hiérarchique sont :

- l'interprétation des résultats : un dendogramme est difficilement exploitable (pertinence de tous les niveaux). Notons aussi que l'arbre obtenu avec une approche par division ou fusion est binaire ;
- les critères d'arrêt sont difficiles à définir et restent limités pour le moment ;
- les noeuds de l'arbre ne sont pas remis en question une fois construits. Les données sont affectées en dur, chaque donnée appartenant à une seule composante ;
- la complexité est élevée : en $O(n^2)$ pour le lien simple et $O(n^2 log(n))$ pour le lien complet et moyen (n est le nombre de données) ;
- les mesures de similarité généralement utilisées sur des données numériques limitent la forme des classes à des formes convexes.

Les avantages de ces algorithmes pour notre application sont les différents niveaux de granularité obtenus. Une telle approche permettrait de fournir des résumés de la collection à l'utilisateur. L'affectation en dur des données aux classes présente néanmoins un problème pour pouvoir proposer un système incrémental. Les algorithmes hiérarchiques classiques sont donc peu adaptés à notre objectif.

2.3.2 Approche par partitionnement direct

Contrairement aux algorithmes hiérarchiques, les algorithmes par partitionnement permettent d'obtenir en sortie une seule classification de données. Cette approche consiste à définir une modélisation des données et ensuite de déterminer les paramètres du modèle. L'ensemble des partitions étant très large, des algorithmes itératifs permettent, en optimisant une fonction d'erreur, de trouver des solutions locales, en itérant entre une phase d'affectation des données aux classes et une phase d'estimation des paramètres. Ces algorithmes permettent d'améliorer petit à petit les classes jusqu'à ce qu'un critère d'arrêt soit vérifié. Ainsi les partitions peuvent être facilement remises en question.

La complexité des modèles dépend de l'approche utilisée. La modélisation probabiliste consiste à faire l'hypothèse que les observations découlent d'une distribution de probabilité. Une composante d'un modèle gaussien est définie par son centre, sa variance et une proportion de mélange. Les algorithmes k-means et k-medoids ont une interprétation géométrique. Ils représentent leurs classes avec des caractéristiques plus simples. Dans le cas du k-means, il s'agit de la moyenne des données de la composante et pour le k-medoids, c'est la donnée la plus représentative de la composante qui la définit. Les paramètres du modèle sont ensuite déterminés en optimisant une fonction d'erreur.

2.3.2.1 Modélisation probabiliste

Cette approche fait l'hypothèse que les données découlent d'une ou de plusieurs lois de distribution de probabilités. Un modèle de mélange contient ainsi plusieurs fonctions de densité, chacune définissant

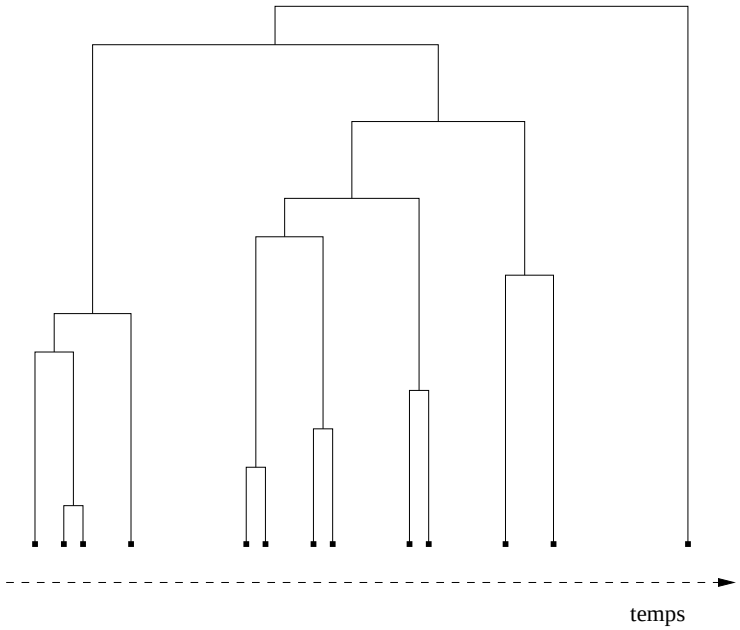


Figure 2.4: Algorithme hiérarchique : exemple de dendogramme obtenu par la méthode simple lien. *La mesure de similarité est basée sur la distance euclidienne.*

une composante du modèle. La densité d'un modèle de mélange est définie par :

$$p(x) = \sum_{k=1}^{K} p_k \cdot p(x|k), \tag{2.2}$$

où K est le nombre de composantes, $p(x|k)$ est la densité de la donnée x générée par la composante k et p_k est la proportion de mélange de la composante k. Dans le cas du modèle de mélange gaussien, le terme $p(x|k)$ est défini par :

$$p(x|k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{\frac{1}{2}}} exp\Big(-\frac{1}{2}(x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k)\Big), \tag{2.3}$$

où Σ_k est la matrice de variance de la composante k, μ_k son centre et d la dimension des données. Notons que selon la distribution utilisée, les paramètres des composantes peuvent présenter des différences.

Les modèles de distribution probabiliste peuvent être paramétrés pour gérer des observations complexes. Plusieurs modèles de mélange existent, présentant des propriétés différentes. Les modèles Gaussiens sont les plus couramment utilisés mais sont sensibles aux données isolées. Un modèle basé sur la loi de distribution de Student [15] est plus robuste face à ce type de données grâce à un degré de liberté à paramétrer agissant sur la forme des composantes.

La méthode la plus connue pour déterminer les paramètres du modèle est celle basée sur le maximum de vraisemblance. Dans la pratique, nous utilisons généralement la log-vraisemblance négative, définie par :

$$L = -\sum_{i=1}^{n} lnp(x_i) = -\sum_{i=1}^{n} ln\Big\{ \sum_{k=1}^{K} p(x_i|k) \cdot p_k \Big\} \tag{2.4}$$

Cette fonction est utilisée comme fonction d'erreur à optimiser. Des algorithmes itératifs sont disponibles dans la littérature pour cette tâche d'optimisation [11].

L'algorithme *Expectation Maximisation* (EM) [37] est une technique classique pour optimiser un tel critère, c'est à dire, déterminer les paramètres du modèle. Il a besoin de valeurs initiales de paramètres et itère ensuite entre deux étapes : l'étape E, où les affectations des données aux composantes sont calculées à partir des paramètres et l'étape M, où les paramètres sont re-calculés en se basant sur les affectations obtenues précédemment. On réalise ces deux étapes jusqu'à convergence de la vraisemblance vers un minimum local. La figure 2.5 ci-après présente un exemple de paramètres déterminés avec l'algorithme EM sur deux jeux de données gaussiennes. Les deux composantes sont initialisées aléatoirement, et au fur et à mesure des itérations, les paramètres convergent vers la bonne solution.

Ce type d'algorithme présente les inconvénients suivants :

- il est dépendant de l'initialisation des paramètres du modèle et fournit ainsi un minimum local de la vraisemblance. Pratiquement, les paramètres sont initialisés aléatoirement ;
- l'optimisation de la log-vraisemblance est réalisée à un nombre constant de composantes. Le nombre de composantes dans le modèle doit être fixé par l'utilisateur.

D'autres algorithmes appartenant à la même famille existent et présentent des propriétés similaires. Nous les présentons dans le chapitre suivant.

La dépendance de ce type d'algorithme aux valeurs initiales des paramètres peut néanmoins être considérée comme un avantage pour fournir un algorithme incrémental. La liberté sur la valeur initiale des paramètres permet de choisir une initialisation appropriée : pour mettre à jour une partition à un temps $t+1$, nous pouvons nous baser sur les paramètres de la partition obtenue au temps t.

Les problèmes d'optimum local et du nombre de composantes peuvent être limités. De nombreuses techniques existent pour améliorer l'optimisation de la vraisemblance, par exemple celle basée sur des sauts aléatoires dans l'espace des paramètres ou la recherche de la meilleure initialisation du modèle (nous y reviendrons dans les chapitres 3 et 5). Et le nombre de composantes dans un modèle peut être déterminé grâce à de nombreux critères statistiques permettant d'évaluer ou de comparer des modèles de mélange (BIC, MDL, ...). Une méthode simple est de faire varier le nombre de composantes entre 1 et K et de sélectionner celui optimisant le critère de comparaison. Une approche automatique basée sur les modèles de mélange est donc possible.

L'approche probabiliste offre plusieurs avantages, à savoir :

- l'affectation dynamique des données, qui permet une grande souplesse pour associer les observations aux composantes. Cette flexibilité présente une propriété intéressante pour concevoir un système incrémental ;
- les partitions obtenues sont facilement interprétables, une méthode simple pour affecter en dur les données étant de les associer à la composante présentant la plus forte probabilité ;
- les algorithmes d'estimation des paramètres dépendent de valeurs initiales sur lesquelles nous pouvons intervenir pour fournir un système incrémental ;
- les partitions peuvent être évaluées et comparées à l'aide de critères statistiques. Ce point est

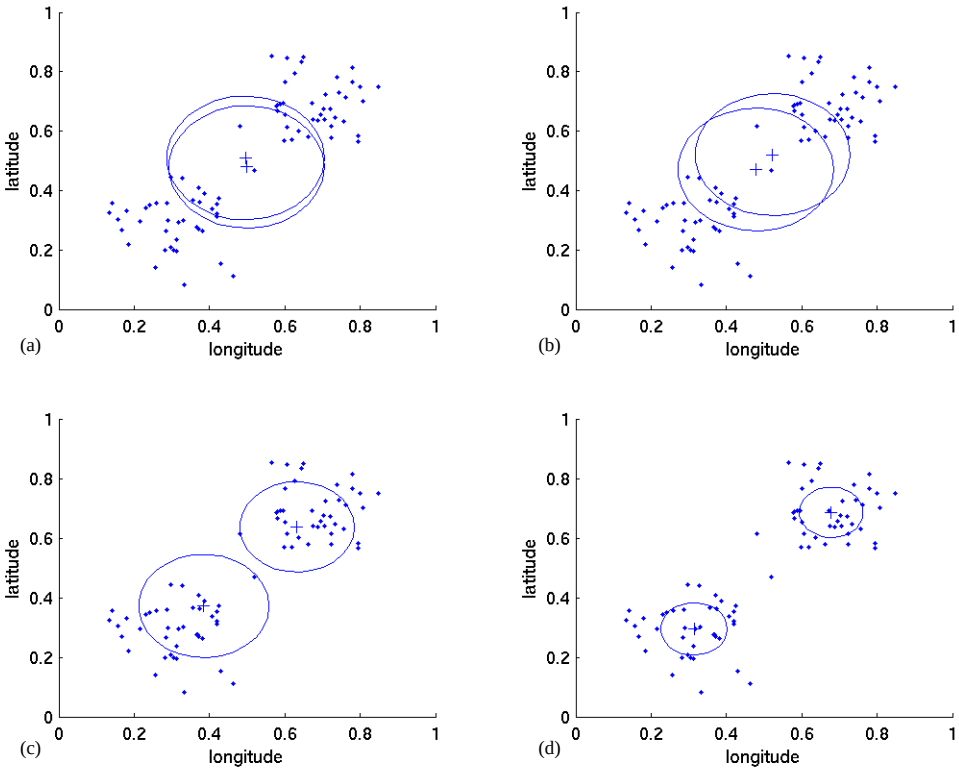


Figure 2.5: Approche par modèle probabiliste : deux jeux de données sont générés à partir de distributions gaussiennes et classés avec un modèle de deux composantes. Nous appliquons l'algorithme EM pour déterminer les paramètres. Les figures (a), (b), (c) et (d) présentent chacune l'état de la partition à une itération donnée lors de l'optimisation de la vraisemblance du modèle. Les cercles et les '+' représentent respectivement les variances et les centres des composantes.

pertinent pour concevoir un système automatique.

Cette approche est généralement utilisée avec des données numériques, les notions de similarité étant plus appropriées pour ce type de données.

Enfin, les modèles probabilistes sont aussi utilisés dans des algorithmes hiérarchiques, par exemple les algorithmes proposés par Fraley [44] et Golberger [53] que nous détaillerons dans un chapitre suivant.

2.3.2.2 Méthode K-means

L'algorithme k-means est l'algorithme le plus utilisé pour la classification de part sa simplicité de mise en oeuvre et son efficacité. Les composantes sont ici représentées par un centre, obtenu en calculant la moyenne de ses observations associées. Cette moyenne favorise l'utilisation de données numériques et entraîne une sensibilité du modèle par rapport aux données isolées. Ce type de données peut avoir un impact important lors du calcul du centre d'une composante. Néanmoins une solution pour obtenir

un algorithme plus robuste à ce type de données consiste à obtenir le centre d'une classe à partir de la médiane de ses données, et non plus de la moyenne.

La fonction à optimiser, appelée *erreur quadratique*, pour déterminer les paramètres se base sur la somme des différences entre un centre et ses observations associées :

$$e^2(x) = \sum_{k=1}^{K} \sum_{i=1}^{n_k} ||x_i^k - \mu_k||^2, \tag{2.5}$$

où x^k est l'ensemble des n_k données associées à la composante k. Ce critère est aussi utilisé avec l'approche probabiliste, l'algorithme k-means dérivant de cette dernière.

Les seuls paramètres à déterminer sont les centres des composantes. L'algorithme k-means, fonctionnant comme l'algorithme EM, est utilisé pour les obtenir. Le nombre de centres K doit être fourni par l'utilisateur et le résultat est dépendant de l'initialisation des centres (comme EM, il permet de trouver un optimum local du critère d'erreur). Contrairement à l'approche probabiliste, les affectations des données sont statiques, chaque donnée étant associée à un seul centre.

Cette approche est dépendante de paramètres arbitraires et présente moins de possibilités que l'approche probabiliste pour pouvoir évaluer ou comparer les modèles obtenus. Une approche automatique est donc plus délicate avec l'approche k-means.

2.3.2.3 Méthode K-medoids

L'approche par *medoids* est similaire à l'approche k-means. Ici, toutefois, la tendance centrale représentant chaque classe est estimée de manière statistiquement robuste : une classe est représentée par sa donnée la plus représentative. Cette définition d'une classe permet d'obtenir un modèle peu sensible aux données isolées et permet de gérer des données ayant des attributs de différents types. Il est néanmoins nécessaire de disposer d'un espace métrique pour comparer les différentes valeurs des attributs.

La fonction à optimiser est similaire à celle utilisée pour l'algorithme k-means et basée sur une notion de distance moyenne des observations à leur *medoid*. Un algorithme itératif, fonctionnant sur le même principe que pour les deux approches précédentes, permet d'optimiser localement ce critère. Comme l'approche k-means, l'affectation des données est statique puisque chaque observation est associée à un seul centre.

Cette approche est dépendante de paramètres arbitraires et le nombre de composantes doit être fourni. Nos données à classer étant numériques, cette approche présente peu d'intérêt en comparaison de l'approche k-means. La seule différence étant les paramètres des composantes qui sont plus adaptés dans notre contexte avec l'approche k-means. La robustesse de l'approche par *medoids* face aux données isolées n'est de plus pas très pertinente pour notre cas d'utilisation. Comme nous le verrons dans la suite, un événement doit pouvoir être composé d'une seule image isolée. Une approche sensible à ce type de donnée est préférable pour notre objectif.

2.3.3 Approche basée sur la densité

Ces algorithmes se basent sur le voisinage des données pour obtenir une partition. Des notions de densité ou de connectivité sont définies pour trouver les plus proches voisins des observations. Une classe est définie par un ensemble d'observations connectées entre elles selon une certaine densité. De nombreux algorithmes utilisent cette méthode, par exemple DBSCAN [42]. Notons que l'approche par grille peut être incluse dans les méthodes basées sur la densité. Elle consiste à diviser l'espace de définition des

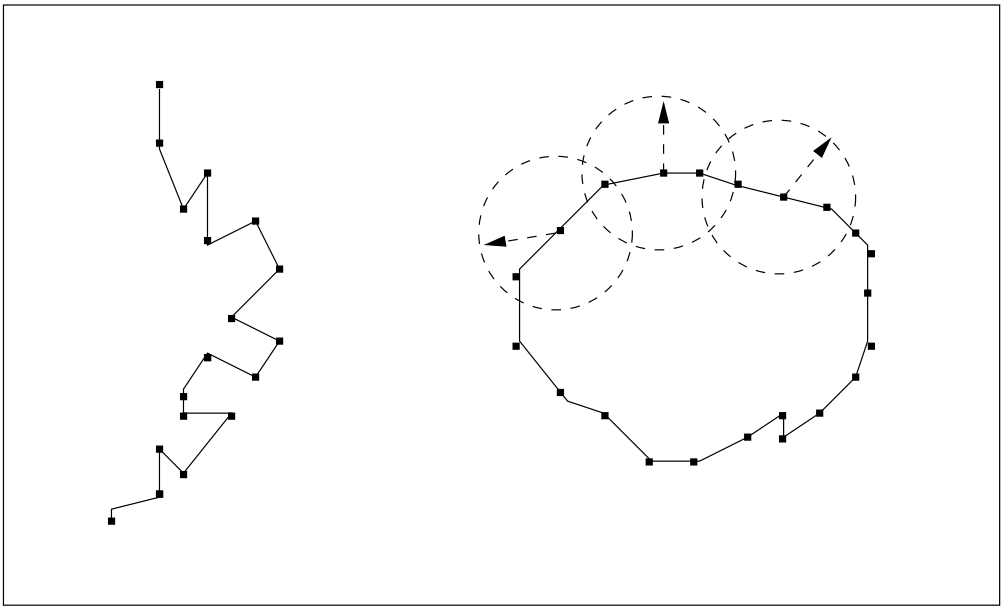


Figure 2.6: Exemple de composantes obtenues avec l'approche par densité : les points représentent les données et les cercles le voisinage des données. Deux données sont connectées si elles peuvent être atteintes via leur voisinage. Les composantes sont les ensembles de données connectées. Aucune contrainte n'est donc posée sur les formes des composantes.

données en petites cellules et à récupérer celles présentant une densité élevée. Les algorithmes DenClue [60] et CLIQUE [1] utilisent cette méthode.

Une fonction de densité simple consiste par exemple à fixer une distance minimale $dist_{min}$ déterminant si deux observations sont voisines. La figure 2.6 ci-après présente un exemple de classification obtenue avec une telle densité. Une classe est composée de données inter-connectées, c'est à dire que pour toute donnée x dans une classe k, il existe une donnée y dans cette même classe telle que $dist(x, y) < dist_{min}$. Des fonctions plus complexes, par exemple une fonction gaussienne, peuvent aussi être utilisées. Ces algorithmes sont généralement utilisés pour classer des observations spatiales [59].

Un avantage important de cette technique est qu'elle permet d'obtenir des composantes de formes variées, difficiles à obtenir avec les algorithmes vus précédemment. Pour les autres techniques, les classes sont limitées aux formes convexes.

Cette approche présente néanmoins quelques désavantages pour notre application. Elle est tout d'abord robuste face aux données isolées et l'interprétation des résultats peut quelquefois poser problème pour des fonctions de densité complexe. Du point de vue description des données, on peut par exemple obtenir une composante contenant deux ensembles d'observations avec des densités différentes. Enfin, la notion de connectivité pose un problème puisqu'elle dépend généralement de paramètres fixés manuellement. Il est donc nécessaire de connaître l'espace de définition des observations pour les initialiser, la notion de voisinage dépendant de l'échelle globale des observations. Voulant éviter de fixer des paramètres arbitraires et ne connaissant pas à l'avance l'espace de définition des données, cette technique nous semble peu appropriée.

2.4 Choix de la famille d'algorithmes de classification au vu des propriétés des données

Avant de choisir la famille d'algorithme la plus appropriée pour notre objectif, nous présentons les caractéristiques des données à classer.

2.4.1 Propriétés des métadonnées temporelles et spatiales

Nous étudions tout d'abord leurs propriétés statiques et ensuite leurs propriétés dynamiques. Les métadonnées peuvent être représentées par un flux de 3-uplet $\{(t,(x,y)) \in \mathbb{R} \times \mathbb{R}^2\}$, ou t est la date et l'heure de prise de vue de l'image, et (x,y) ses coordonnées géographiques.

Une collection est composée de plusieurs événements temporels, définis sur des intervalles de temps variables : cela peut aller de quelques secondes (des images isolées) à plusieurs semaines (vacances). Les classes spatiales sont définies sur des espaces plus ou moins grands, allant de plusieurs mètres (dans une pièce) à plusieurs centaines de kilomètres (photographies de vacances). Pour la classification spatiale, les formes des classes peuvent être variées. Une classe regroupant des images prises lors d'une balade peut avoir une forme en courbe. Néanmoins, nous avons noté, à partir des coordonnées géographiques d'une collection réelle, que les classes spatiales étaient généralement compactes et concentrées en des lieux précis.

Chaque classe d'images, temporelle ou spatiale, peut contenir un nombre variable d'images, pouvant aller de 1 à plusieurs centaines. Un lieu habituel contiendra beaucoup d'images alors que certain lieux de vacances visités une seule fois en contiendront peu. Pour les classes temporelles, un événement associé à des vacances comprendra un grand nombre d'images, mais il est aussi possible d'obtenir des images isolées temporellement. Aucun à priori ne peut donc être fixé sur le nombre de données par classe.

Du point de vue dynamique, le processus de génération de la collection peut entraîner un problème de mise à jour des partitions. Rappelons tout d'abord que les utilisateurs prennent leurs images par paquets [81]. Un lien peut alors être établi entre deux données successives pour l'affectation aux classes (temporelles ou spatiales). On observe une dépendance entre les données et on peut en déduire que : $P(d_t \in k | d_{t-1} \in k') \neq P(d_t \in k)$ où $P(d_t \in k)$ est la probabilité d'affectation de la donnée d prises au temps t à la classe k. De plus grands efforts de ré-organisation des classes sont ainsi nécessaires que si les données d_t et d_{t-1} étaient totalement indépendantes. Par exemple, au fur et à mesure que les données d'un même lieu sont ajoutées, les classes sont optimisées localement sur cet endroit spécifique, elles peuvent donc être plus difficiles à modifier si un nouveau lieu apparaît.

Chaque composante, spatiale ou temporelle, peut donc présenter des propriétés variables. Le problème de classification de ce type de données n'est donc pas simple et nous devons faire face à des problèmes de classifications classiques :

- la gestion des petits échantillons ;
- la gestion de données non structurées ;
- enfin, pour la classification spatiale, la forme des classes variée.

2.4.2 Approche par modèle de mélange

Les classifications doivent être réalisées automatiquement sans avoir à régler manuellement de paramètres gênants. L'algorithme de classification doit être automatique. En particulier, il faut pouvoir déterminer le nombre de composantes à partir de la structure des données. Parmi les approches explicitées, l'approche par modèle de mélange semble la plus appropriée sur ce point. En effet, les partitions peuvent être évaluées avec des critères statistiques, permettant de comparer des modèles avec des hypothèses différentes, à un faible coût.

L'aspect incrémental nécessite une technique où les affectations des données aux composantes sont très flexibles. Comme explicité dans la section précédente, les partitions doivent pouvoir être remises régulièrement en question (au vu de leur propriété dynamique). Là encore les modèles de mélange semblent les plus adaptés: l'affectation des données aux composantes est très souple. Ce point permet de facilement modifier les paramètres du modèle, et donc les affectations des données aux composantes, après l'ajout d'une nouvelle donnée dans la classification. Les algorithmes d'estimation des paramètres présentent de plus la propriété d'avoir besoin de valeurs initiales de paramètres et nous avons vu que ce point est un avantage pour concevoir un système incrémental. Le couple *modèle de mélange* et *algorithme de type EM* semble approprié pour notre objectif.

Les modèles de mélange fournissent de plus des partitions facilement interprétables et ce avec une complexité raisonnable. La gestion des petits échantillons dépendra du modèle choisi pour représenter les données. Comme nous le verrons dans le chapitre suivant, plusieurs modèles existent présentant des propriétés différentes sur ce type de données.

Enfin, l'aspect hiérarchique peut aussi être géré avec les modèles de mélange. Il existe dans la littérature plusieurs méthodes pour construire des classifications hiérarchiques à partir d'un modèle de mélange. Notre solution consiste dans un premier temps à développer un algorithme incrémental fournissant en sortie une partition des données. Ensuite, nous combinerons cet algorithme à un algorithme hiérarchique pour obtenir nos classifications finales.

Le seul désavantage du choix sur les modèles de mélange est lié à la forme des classes spatiales. Ce choix entraîne des classes de formes convexes alors que dans la réalité les formes sont diverses. Néanmoins, une approche basée sur la densité, qui nous semble la plus adaptée pour régler ce problème, n'est pas possible dans notre contexte. L'ensemble des données à classer n'étant pas connu, il paraît difficile de définir une notion de connectivité entre les données.

L'approche probabiliste semble appropriée pour développer un système hiérarchique et incrémental, sans dépendre de paramètres arbitraires. Nous opterons ainsi pour une approche basée sur les modèles de distribution gaussien. Le choix de ce modèle sera motivé dans le chapitre suivant où plusieurs modèles sont présentés.

2.5 Conclusion

Nous objectif est de fournir un système automatique pour organiser une collection d'images personnelles. Notre approche consiste à construire deux classifications spatiale et temporelle distinctes à partir des métadonnées de chaque image. Une indépendance entre les partitions permet ensuite plusieurs solutions pour les combiner en une seule partition hybride.

Nous avons choisi de traiter ce problème comme une tâche de classification automatique. Afin de fournir un système adapté, nous proposons que les classifications soient :

- incrémentales, afin de prendre en compte le processus de construction de la collection ;
- hiérarchiques afin de pouvoir résumer la collection pour faciliter son parcours sur un terminal mobile.

Les différentes techniques de classification ont été abordées dans ce chapitre afin de mettre en valeur la plus pertinente pour notre objectif. Nous avons choisi une approche basée sur les modèles de mélange. Celle-ci paraît la plus appropriée pour fournir un système incrémental (affectation souple des données aux classes) et automatique (existence d'algorithmes d'estimation des paramètres de complexité raisonnable et de critères statistiques pour comparer les modèles). Le prochain chapitre présente la modélisation de notre système, où le choix sur les modèles gaussiens est motivé.

CHAPITRE 3

Modèle de mélange probabiliste pour les données spatiales et temporelles

Nous présentons dans ce chapitre la modélisation de nos métadonnées spatiales et temporelles, obtenues lors de la prise de vue de chaque image. Nous motivons notre choix sur le modèle de mélange gaussien et l'utilisation de l'algorithme EM pour fournir notre propre algorithme d'estimation des paramètres, décrit dans le chapitre 5. Enfin, nous avons vu dans le chapitre précédent que certaines classes peuvent être associées à de petits échantillons de données. Ce type de structure pose des problèmes lors de l'estimation des paramètres. Nous présentons une technique pour la gestion des petits échantillons.

3.1 Introduction

L'objectif de ce chapitre est de présenter la modélisation de nos métadonnées spatiales et temporelles et de fournir les outils nécessaires pour leur classification. Nous avons opté pour une modélisation à l'aide de modèle de mélange et une procédure de classification statistique. Les avantages d'une telle approche ont été explicités dans le chapitre précédent.

Nous supposons, avec cette approche, que les données sont tirées aléatoirement d'une densité de probabilité. Une classification peut ainsi être modélisée par une combinaison linéaire de densités simples : l'ensemble des classes, appelées composantes du mélanges et associées chacune à une densité de probabilité, représente le modèle de mélange. Un point important est le choix de la densité : plusieurs lois de distribution sont utilisées dans le cadre de la classification de données, chacune présentant des propriétés différentes (capacité de modélisation, difficultés sur l'estimation des paramètres du modèle de mélange). Après avoir rappelé la définition et les propriétés d'un modèle de mélange, nous présentons plusieurs lois élémentaires et validons notre choix sur la distribution gaussienne.

Nous introduisons ensuite les paramètres du modèle gaussien, avec les différents choix de contraintes à priori possibles. Ces contraintes concernent les matrices de variance des composantes, ayant un impact sur leur géométrie, et les proportions du mélange. Ce choix est défini en fonction des propriétés des données spatiales et temporelles à classer.

L'estimation des paramètres d'un modèle de mélange est un problème complexe. Les algorithmes existants nécessitent de disposer de la complexité du modèle (nombre de composantes) et sont dépendants de paramètres fixés manuellement. Nous présentons dans ce chapitre les algorithmes les plus employés

pour les estimer : les algorithmes EM, SEM et SAEM, et CEM et CAEM. Les propriétés de chacun sont mises en valeur (convergence, complexité,...).

Enfin, un problème courant en classification est la gestion des *petits échantillons*. Nous parlons de *petit échantillon* quand une composante est associée à un faible nombre de données. L'estimation des paramètres peut être biaisée ou même impossible avec ce type de données. L'approche pour régler ce problème consiste à régulariser les matrices de variance. Après avoir présenté les différents procédés existants, nous définissons notre propre régularisation.

3.2 Le modèle de mélange

Les modèles de mélange permettent d'obtenir des modèles complexes en s'appuyant sur plusieurs fonctions de densités de probabilités. Ces fonctions sont appelées les composantes du mélange. Avant de présenter les lois élémentaires, nous rappelons la définition d'un modèle de mélange.

3.2.1 Définition

Les densités de mélange correspondent à une combinaison linéaire de densités simples. Soient X et Z deux variables aléatoires, $\mathcal{N}(x|z)$ la densité de X conditionnellement à Z et $h(z)$ la densité de Z. La densité de mélange est définie par :

$$f(x) = \int \mathcal{N}(x|z)h(z)dz \tag{3.1}$$

Quand Z est défini sur un ensemble fini de valeurs $\{1, \ldots, K\}$, la distribution de Z est discrète et affecte une probabilité positive à chacune de ces valeurs. On parle de densité de mélange fini :

$$f(x) = \sum_{k=1}^{K} p_k.\mathcal{N}_k(x) \tag{3.2}$$

avec

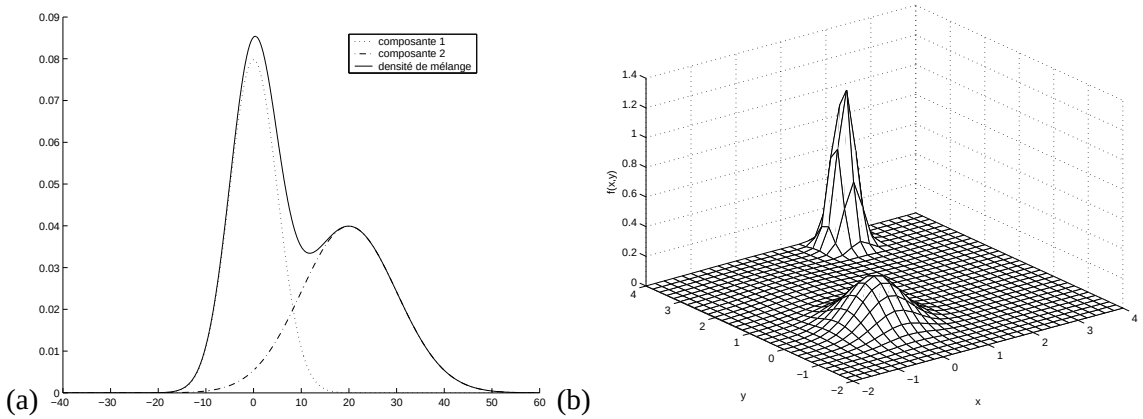
$$\mathcal{N}_k(x) = \mathcal{N}(x|Z = k) \text{ et } p_k = P(Z = k), \tag{3.3}$$

où les p_k sont les proportions de mélange vérifiant les contraintes suivantes : $0 \leq p_k \leq 1$ pour tout $k = 1, \ldots, K$ et $p_1 + \cdots + p_K = 1$.

Dans notre contexte, le nombre de classes à obtenir dans une classification est limité. Nous nous intéressons donc dans la suite aux densités de mélange fini.

3.2.2 Modèle de mélange pour la classification

Ce paragraphe présente plusieurs distributions de probabilité proposées dans la littérature pour un objectif de classification. Les définitions et propriétés de trois lois de probabilité sont détaillées : la distribution gaussienne, la distribution de Student et la distribution de Dirichlet.



(a) (b)

Figure 3.1: Exemple de densités gaussiennes $f(x)$ et $f(x, y)$, respectivement représentées par les figures (a) et (b).

3.2.2.1 La loi gaussienne

La loi normale joue un rôle central dans la théorie de probabilités et dans ses applications statistiques. Dans la situation multivariée, la variable aléatoire X est définie dans l'espace euclidien $\mathbb{R}^d$ et la densité de la composante k ($1 \leq k \leq K$) est définie par :

$$\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{\frac{1}{2}}} exp\Big(-\frac{1}{2}(x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k)\Big), \tag{3.4}$$

où μ_k est la moyenne de la composante k ($\mu_k \in \mathbb{R}_d$) et Σ_k sa matrice de variance de dimension $d \times d$, symétrique et définie positive.

La densité de mélange est donc définie par :

$$f(X) = \sum_{k=1}^{K} p_k . \mathcal{N}(X|\mu_k, \Sigma_k), \tag{3.5}$$

où les probabilités p_k sont les proportions de mélange.

Les figures 3.1(a) et (b) ci-dessus présentent des exemples de modèles de mélange gaussien, respectivement en dimensions 1 et 2. Les deux mélanges sont chacun composés de deux composantes avec des variances différentes. Les composantes gaussiennes ont des formes symétriques.

L'estimation des paramètres d'un modèle gaussien est peu robuste aux petits échantillons de données, le centre d'une composante étant obtenu à partir de la somme des données pondérées par les probabilités d'affectation *a posteriori* [93]. Dans la suite, nous définissons qu'une telle distribution est *sensible* aux données isolées (contrairement à *robuste*). La figure 3.2 ci-après présente un exemple d'une composante obtenue avec l'algorithme EM (détaillé dans la suite) sur un jeux de données comprenant un petit échantillon isolé (situé en $[6; 6]$). Le centre de la composante varie fortement vers l'ensemble des données isolées. Ces dernières ont beaucoup de poids lors de la phase d'estimation des paramètres. Le modèle obtenu est peu pertinent pour représenter la structure des données. Un modèle plus robuste aurait plus centré la composante sur le groupe de données situé sur la gauche.

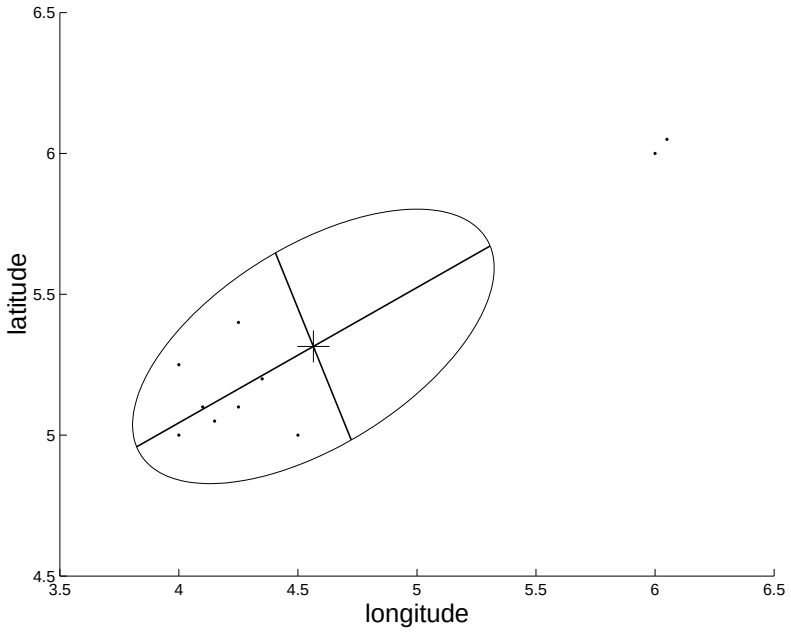


Figure 3.2: Sensibilité de la densité gaussienne face aux données isolées. Les points représentent les données et le '+' et l'ellipse sont respectivement le centre et la variance de la composante.

Cette sensibilité est généralement considérée comme un inconvénient puisque dans de nombreux problèmes de classification les données sont bruitées, comprenant des observations peu structurées et isolées. Dans notre contexte, cela devient un avantage quand nous comparerons des modèles de complexités différentes. La sensibilité aux données isolées a un impact sur la mesure de qualité du modèle (la vraisemblance, que nous présentons dans la suite) : sur la figure 3.2, le modèle représente mal les données, entraînant une mesure de qualité plus faible que si la composante était centrée sur les données de gauche. Quand on comparera plusieurs solutions de modèles, cette sensibilité va donc désavantager les modèles ayant des données isolées associées à des composantes éloignées.

3.2.2.2 La loi de Student

La loi de Student est une généralisation de la loi gaussienne. Elle présente l'avantage sur cette dernière de disposer d'un paramètre supplémentaire permettant de jouer plus finement sur la forme de chaque composante. La valeur de ce paramètre permet de régler la robustesse du modèle face aux petits échantillons. Dans la littérature, ce modèle est essentiellement présenté comme une alternative au modèle gaussien face à ce type de donnée [15, 88, 93, 102].

La variable aléatoire x suit une loi de Student si sa fonction de densité est de la forme :

$$f(x) = \frac{\Gamma(\frac{v+1}{2})}{\sqrt{v\pi}\Gamma(\frac{v}{2})}(1 + \frac{t^2}{v})^{-\frac{v+1}{2}}, \tag{3.6}$$

où Γ est la fonction *Gamma* et v le nombre de degrés de liberté.

La figure 3.3(a) ci-après présente un exemple d'une composante suivant la loi de distribution de Student pour plusieurs valeurs du degré de liberté. La distribution de Student est symétrique par rapport à l'axe des ordonnées et converge vers la distribution gaussienne quand $v \to \infty$.

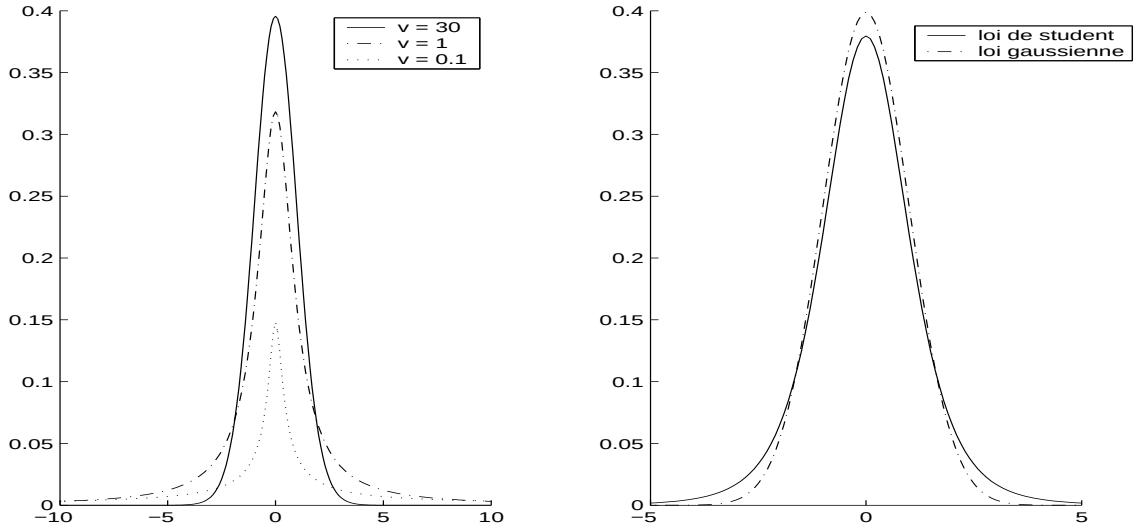


Figure 3.3: (a) Exemples de la densité de Student pour plusieurs degrés de liberté à paramètres (μ, Σ) constants. (b) Comparaison de la densité de Student et de la densité gaussienne.

La figure 3.3(b) ci-après compare la densité de Student à la densité gaussienne. Nous remarquons que pour les deux valeurs de v, la densité de Student a une base plus large qu'une gaussienne. Cette propriété à un impact sur les modèles obtenus : ils sont plus robustes face aux données isolées. Celles-ci ont moins de poids que dans le cas du modèle gaussien : le degré de liberté v permet de limiter le poids des données isolées pendant le calcul des paramètres du modèle de mélange. Plus v est petit, plus le modèle est robuste face à ce type de données.

Un premier problème réside néanmoins dans l'estimation du paramètre v, qui doit être calculé itérativement en résolvant une équation nécessitant la dérivée de tous les termes liés aux degrés de liberté dans la fonction de densité [93]. L'estimation de ce paramètre présente ainsi une forte complexité en temps de calcul. Une technique pour éviter ce problème est de fixer une valeur pour toutes les composantes : cela revient à régler manuellement la robustesse du modèle face aux petits échantillons.

Enfin, les travaux de Shoham [102] ont fait apparaître des problèmes lors de l'estimation des paramètres de cette distribution pour un degré de liberté inférieur à 10. Il semblerait que la base plus large de la distribution de Student empêche les composantes de bouger hors des régions présentant de fortes densités. Ce problème a aussi été rencontré lors de nos expériences.

3.2.2.3 La distribution de Dirichlet

La distribution multinomiale de Dirichlet est une loi de distribution continue, généralisant la loi de distribution beta. Son utilisation est généralement liée aux problèmes dont les variables aléatoires sont comprises dans un intervalle $[0; 1]$, ce qui n'est pas notre cas puisque les données à classer sont comprises dans $\mathbb{R}^+$. Bouguila et al. [16] ont proposé son adaptation pour le problème de classification.

Si une variable aléatoire x suit une loi de Dirichlet alors sa densité jointe est donnée par :

$$p(x_1, \ldots, x_d) = \frac{\Gamma(|\alpha|)}{\prod_{i=1}^{d+1} \Gamma(\alpha_i)} \prod_{i=1}^{d+1} x_i^{\alpha_i - 1}, \tag{3.7}$$

où

$$\begin{aligned} \sum_{i=1}^{d} x_i &< 1 \\ 0 < x_i < 1 \quad \forall i \in [1,d] & \\ x_{d+1} &= 1 - \sum_{i=1}^{d} x_i \\ |\alpha| &= \sum_{i=1}^{d+1} \alpha_i \\ \alpha_i > 0 \quad \forall i \in [1,d+1] & \end{aligned}$$

La distribution de Dirichlet avec un vecteur de paramètre $\overrightarrow{\alpha} = (\alpha_1, \ldots, \alpha_{d+1})$ peut être représentée comme une distribution dans le simplex $A_d = \{(x_1, \ldots, x_d), \sum_{i=1}^{d} x_i < 1\} \in \mathbb{R}^+$. Ce simplexe est un réel handicap puisque les données peuvent ne pas y appartenir : nos métadonnées spatiales ou temporelles sont définies dans $\mathbb{R}^+$. Bouguila et al. [16] ont proposé la distribution généralisée de Dirichlet (GDD) pour contourner le problème. Si le vecteur aléatoire $\overrightarrow{x} = \{x_1, \ldots, x_d\}$ suit une GDD avec un vecteur de paramètres $\overrightarrow{\alpha} = (\alpha_1, \ldots, \alpha_{d+1})$, la fonction de densité jointe est définie par :

$$p(x_1, \ldots, x_d) = \frac{\Gamma(|\overrightarrow{\alpha}|)}{A^{|\overrightarrow{\alpha}|-1} \prod_{i=1}^{d+1} \Gamma(\alpha_i)} \prod_{i=1}^{d+1} x_i^{\alpha_i - 1}, \tag{3.8}$$

où $x_{d+1} = A - |\overrightarrow{x}|$.

La distribution de Dirichlet généralisée est très flexible et permet de multiples formes symétriques et asymétriques. Ce point est un avantage pour la classification des métadonnées spatiales, les formes des classes pouvant être variées.

L'inconvénient de cette distribution vient de la technique d'estimation des paramètres $\overrightarrow{\alpha}$. L'algorithme proposé par Bouguila [16] utilise la matrice d'information de Fisher, entraînant une complexité plus élevée pour estimer les paramètres que les algorithmes utilisés avec la distribution gaussienne ou de Student (avec v fixé).

3.2.2.4 Discussion sur le choix du modèle de mélange

Le choix de la loi dépend de 2 contraintes liées à notre objectif :

1. les classifications obtenues doivent être cohérentes avec le souhait d'un utilisateur ;

2. la complexité en temps de calcul de l'algorithme d'estimation des paramètres doit être raisonnable. Nous rappelons que notre système doit fonctionner sur un appareil mobile, ayant une puissance de calcul limitée.

Nous avons vu dans le chapitre 2 qu'un événement ou un lieu peuvent être composés de peu d'images. Une classe doit pouvoir être associée à de petits échantillons de données. Une donnée isolée doit par exemple pouvoir entraîner la création d'une composante (section 2.4.1, page 30). Notre loi de distribution doit ainsi être sensible à ce type de données.

Dans notre contexte, le modèle de Student ne présente donc pas d'avantages par rapport au modèle gaussien. Pour que les petits échantillons de données soient convenablement retrouvés, il est nécessaire de fixer le degré de liberté v à une valeur élevée. Or dans ce cas, la distribution de Student est similaire à la distribution gaussienne.

Notre choix se porte donc sur la distribution gaussienne ou de Dirichlet. L'algorithme d'estimation des paramètres devant présenter une complexité minimum, notre préférence va à la distribution gaussienne. Comme nous l'avons vu, la distribution de Dirichlet présente un algorithme d'estimation des paramètres plus complexe que pour la distribution gaussienne. Nous optons sur le modèle gaussien, ce choix étant motivé par deux avantages par rapport aux autres modèles présentés :

- des algorithmes efficaces existent, de complexité raisonnable, permettant de déterminer les paramètres du modèle ;
- une sensibilité aux données isolées, un atout pour la gestion des petits échantillons.

Avant de présenter les algorithmes d'estimation des paramètres, nous présentons en détail le modèle gaussien.

3.3 Les paramètres du modèle gaussien

Les modèles gaussiens correspondent à un ensemble de contraintes sur les matrices de variance et sur les proportions de mélange. Le choix des contraintes à une influence sur les classifications voulues en sortie. Nous présentons 28 modèles différents en utilisant une décomposition spectrale des matrices de variance. Une composante k du modèle est définie par :

- un centre μ_k, défini par la moyenne des données associées à la composante ;
- une matrice de variance Σ_k, représentant la dispersion des données autour de son centre ;
- une proportion de mélange p_k, définissant la fraction des données appartenant à cette classe.

Nous présentons les différents types de contraintes à priori et nous motivons ensuite nos choix de contraintes.

3.3.1 Contraintes sur les paramètres du modèle

Une connaissance à priori sur les observations à classer peut permettre de fixer des contraintes à priori sur les paramètres du modèle de mélange. Par exemple, si l'utilisateur sait que chaque classe contient autant d'observations, les proportions de mélange doivent être égales. Pour les matrices de variance, les contraintes portent sur la géométrie des composantes. Quatre types de contraintes existent :

- modèle linéaire sphérique : Σ_k est de la forme $\Sigma_k = \sigma^2 I$ où σ est libre et I une matrice identité ;
- modèle linéaire diagonal : Σ_k est de la forme $\Sigma_k = diag(\sigma_1^2, \ldots, \sigma_d^2)$ où $(\sigma_1^2, \ldots, \sigma_d^2)$ est libre et $diag(\sigma_1^2, \ldots, \sigma_d^2)$ est une matrice diagonale ;

- modèle linéaire général : $\Sigma_1 = \cdots = \Sigma_k = \Sigma$, où Σ est une matrice symétrique définie positive ;
- modèle quadratique : aucune contrainte sur les matrices de variance Σ_k n'est posée.

Poser des contraintes à priori sur un modèle de mélange permet de mieux estimer les paramètres θ et d'éviter une surparamétrisation. Une surparamétrisation est obtenue quand le nombre de paramètres libres est trop grand par rapport aux nombres d'observations à classer. Dans un tel cas, l'estimation des paramètres n'est pas fiable.

Décomposition des matrices de variances

Banfield et Raftery [7] ont proposé une généralisation des différents types de matrices de variance présentés précédemment. Ils utilisent une décomposition spectrale de la matrice, qui a été redéfinie ensuite par Celeux et Govaert [29] afin de simplifier certains calculs pour l'estimation. La décomposition spectrale de la matrice de variance de la classe k est définie par :

$$\Sigma_k = \lambda_k \mathbf{D}_k \mathbf{A}_k \mathcal{D}'_k, \tag{3.9}$$

où :

- $\lambda_k = |\Sigma_k|^{1/d}, \lambda_k \in \mathbb{R}^*_+$;
- $\mathbf{D}_k$ est la matrice orthogonale des vecteurs propres de Σ_k ;
- $\mathbf{A}_k = diag(a_{1k}, \ldots, a_{dk})$ est la matrice diagonale des valeurs propres normalisées avec les a_{ik} classés par ordre décroissant sur la diagonale et $|A_k| = 1$.

L'avantage de cette décomposition est son unicité et sa facilité d'interprétation du point de vue géométrique : λ_k représente le volume de la classe, $\mathbf{D}_k$ son orientation et $\mathbf{A}_k$ sa forme.

Nous dénombrons 14 modèles de matrices de variance pouvant être regroupés en trois grandes familles :

- sphérique : la matrice de la classe $\mathbf{A}$ est la matrice identité, $A = I$. La matrice d'orientation D n'a aucun impact sur les paramètres ;
- diagonale : la matrice $B = DAD'$ est diagonale, D est une matrice de permutation de la base canonique commune à toutes les composantes ;
- générale : cette famille regroupe tous les autres modèles. Nous notons $C = DAD'$.

Le tableau 3.1 ci-dessus résume les différents modèles de matrices et détaille le nombre de paramètres libres pour chaque configuration. Le nombre de paramètres libre met en valeur la complexité de chaque modèle.

Afin de mieux comprendre la signification de cette décomposition de la matrice de variance, nous proposons un exemple. La figure 3.4 ci-après présente une interprétation géométrique d'une matrice de

Famille	Modèles	Nombre de paramètres
Sphérique	$[\lambda I]$	$a+1$
	$[\lambda_k I]$	$a+K$
Diagonale	$[\lambda B]$	$a+d$
	$[\lambda_k B]$	$a+d+K-1$
	$[\lambda B_k]$	$a+Kd-(K-1)$
	$[\lambda_k B_k]$	$a+Kd$
Générale	$[\lambda C]$	$a+b$
	$[\lambda_k C]$	$a+b+K-1$
	$[\lambda D A_k D']$	$a+b+(K-1)(d-1)$
	$[\lambda_k D A_k D']$	$a+b+(K-1)d$
	$[\lambda D_k A D'_k]$	$a+Kb-(K-1)d$
	$[\lambda_k D_k A D'_k]$	$a+Kb-(K-1)(d-1)$
	$[\lambda C_k]$	$a+Kb-(K-1)$
	$[\lambda_k C_k]$	$a+Kb$

Table 3.1: Les différents modèles de matrices de variance, où $a = Kd + K - 1$ si les proportions sont libres et $a = Kd$ sinon, et $b = (d\frac{(d+1)}{2})$.

variance en dimension 2. Les matrices A_k et D_k sont exprimées par :

$$A_k = \begin{pmatrix} a_k & 0 \\ 0 & 1/a_k \end{pmatrix}, D_k = \begin{pmatrix} \cos\rho_k & -\sin\rho_k \\ \sin\rho_k & \cos\rho_k \end{pmatrix}, \tag{3.10}$$

où ρ_k est l'angle de rotation.

Contraintes sur les proportions de mélange

Enfin, le nombre de modèles passe de 14 à 28 si on prend en compte les contraintes sur les proportions de mélange. Deux configurations remarquables sont possibles :

- les proportions de mélange sont identiques pour toutes les composantes ($p_1 = \cdots = p_K$) ;
- les proportions sont libres et doivent être estimées.

3.3.2 Choix des paramètres de modélisation

Le modèle gaussien permet donc plusieurs types de contraintes sur les paramètres. La modélisation de notre système nécessite donc de choisir quels types de contraintes sont à même de fournir les meilleurs résultats.

Nous avons vu dans le chapitre précédent que, dans notre contexte, les classes spatiales à obtenir peuvent avoir des formes diverses. Il est nécessaire de disposer de composantes avec des formes les plus flexibles possibles. Notre choix se porte donc sur le modèle le plus souple, présentant le plus de liberté : le modèle général $[\lambda_k C_k]$. Le volume, l'orientation et la géométrie des composantes sont libres

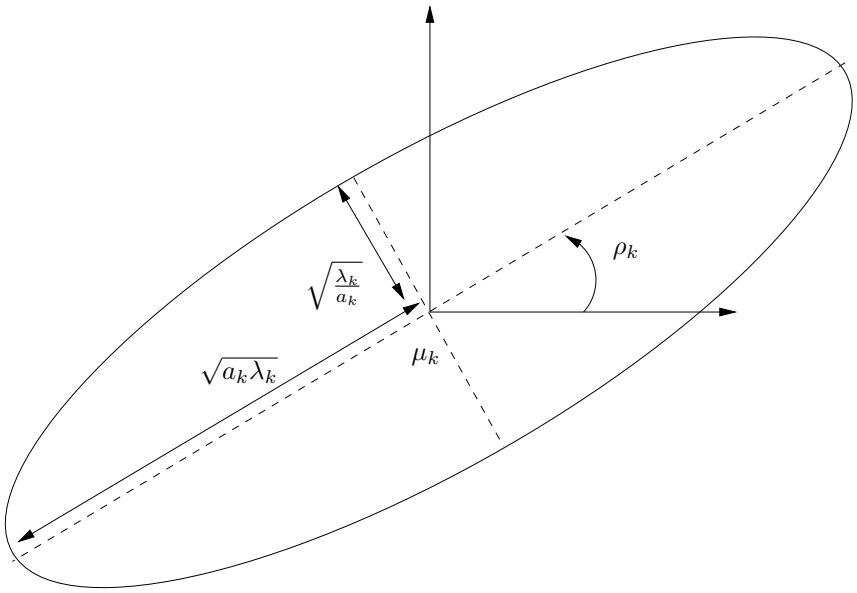


Figure 3.4: Illustration géométrique d'un modèle de mélange gaussien en dimension 2.

et indépendants pour chaque composante. Ces contraintes ont surtout un impact pour les métadonnées spatiales. Pour les métadonnées temporelles, on ne peut paramétrer que le volume des composantes puisque la dimension de ces données est de 1. Une liberté de forme et d'orientation n'est pas défini pour cette dimension.

Enfin, puisque les classes peuvent contenir un nombre variable d'images, les proportions de mélange doivent être libres. Nous ne pouvons pas imposer dans notre contexte que chaque classe ait le même nombre d'images. En conclusion, les contraintes à priori sont :

- des matrices de variance de modèle générale $[\lambda_k C_k]$;
- des proportions de mélange p_k libres.

3.4 Critères d'optimalité et algorithmes d'estimation des paramètres

Nous présentons dans cette partie plusieurs algorithmes permettant de déterminer les paramètres d'un modèle de mélange gaussien. Nous proposons deux classes d'algorithmes, la première étant basée sur une approche mélange (estimation des paramètres à partir des données) et l'autre sur une approche mixte (estimation simultanée des paramètres θ et des étiquettes d'affectation z). Ces deux approches diffèrent par le critère d'erreur à optimiser : la vraisemblance pour l'approche mélange et la vraisemblance classifiante pour l'approche mixte.

Avant d'aborder les classes d'algorithmes, nous présentons le critère de maximum à posteriori (MAP), que nous utilisons pour obtenir l'affectation binaire des données aux composantes. Ce critère nécessite

de connaître les paramètres θ du modèle. Ces paramètres étant connus, nous pouvons ensuite affecter les observations suivant la probabilité *a posteriori* la plus probable. La probabilité *a posteriori* que x_i soit associé à la classe k se calcule par le théorème de Bayes :

$$t_{ik} = P(z_i = k|X, \theta) = \frac{p_k \cdot \mathcal{N}(x|\theta_k)}{\sum_{k=1}^{K} p_k \cdot \mathcal{N}(x|\theta_k)}$$

Cette méthode est simple et puise sa justification en théorie bayésienne (pour plus de détails, voir [11]). Nous affectons une donnée i à la composante avec le terme t_{ik} maximum. Nous notons t la matrice comprenant les termes t_{ik}, composée de n lignes et K colonnes. Chaque terme $t_{ik} \in [0, 1]$ est la probabilité *a posteriori* que la donnée i soit générée par la composante k. La somme des lignes est de 1 ($\sum_{k=1}^{K} t_{ik} = 1$).

3.4.1 Critère du maximum de vraisemblance et algorithmes pour l'optimiser

Nous rappelons la définition du maximum de vraisemblance et nous détaillons ensuite plusieurs algorithmes d'estimation : les algorithmes Expectation-maximisation (EM), Stochastic EM (SEM) et stochastic Annealing EM (SAEM).

3.4.1.1 Définition du maximum de vraisemblance

Soient n observations $\in X$ indépendantes, la densité de $x_1, \ldots, x_n$ peut être exprimée par le produit de toutes les densités marginales :

$$f(X|\theta) = f(x_1, \ldots, x_n|\theta) \tag{3.11}$$

$$= \prod_{i=1}^{n} f(x_i|\theta) \tag{3.12}$$

Cette densité est appelée la vraisemblance des paramètres θ. On utilise plus généralement pour des raisons de calcul le logarithme népérien de la vraisemblance négative défini par :

$$L(\theta) = -\sum_{i=1}^{n} ln\Big(\sum_{k=1}^{K} p_k \cdot \mathcal{N}(x_i|\theta_k)\Big), \tag{3.13}$$

où θ_k représente les paramètres de la composante k.

L'estimation des paramètres par maximum de vraisemblance est la méthode la plus courante. Elle consiste à choisir les paramètres du modèle qui minimisent $L(\theta)$. Les estimateurs obtenus avec cette méthode présentent de bonnes propriétés statistiques :

- ils convergent en probabilité vers les vraies valeurs des paramètres ;
- ils sont asymptotiquement sans biais et gaussiens.

Néanmoins, le maximum de vraisemblance n'est pas borné. Si on prend autant de classes que d'observations et que chaque donnée est associée au centre d'une classe, la vraisemblance diverge vers l'infini si la variance de chaque classe tend vers zéro. Enfin l'optimum de la vraisemblance est rarement réalisable analytiquement et nécessite de recourir à des méthodes itératives. Nous présentons dans la suite des algorithmes permettant d'optimiser la vraisemblance.

3.4.1.2 L'algorithme Expectation-Maximisation (EM)

L'algorithme EM [37] est la méthode la plus employée pour estimer les paramètres d'un modèle de mélange. De multiples variantes de cette technique fondamentale sont présentées dans la suite.

Supposons que nous observons les valeurs d'un ensemble X de n variables aléatoires, mais pas les valeurs d'un autre ensemble Z et qu'en se basant sur ces données, nous voulons trouver la vraisemblance maximale estimée d'un modèle pour X et Z. Ce problème ne peut pas être résolu directement, et le problème correspondant avec l'ensemble Z connu serait plus facile. Nous supposons que l'ensemble des valeurs de Z est fini, défini sur $\{1, \ldots, K\}$.

La probabilité jointe de X et Z est paramétré, en utilisant θ, par $P(z, x|\theta)$. La probabilité marginale pour X est alors $P(x|\theta) = \sum_{i=1}^{n} P(z_i, x|\theta)$. Soit une valeur observée x, nous voulons trouver la valeur de θ qui optimise la log-vraisemblance négative $L(\theta) = -ln\ P(x|\theta)$.

Le principe de l'algorithme EM consiste à construire une séquence d'estimation de paramètres θ^0, …, θ^m, sur laquelle la log-vraisemblance négative suit une décroissance monotone (pour tout m, $L(\theta^{m+1}) \leq L(\theta^m)$). L'idée est de choisir les paramètres θ^{m+1} à partir de θ^m qui optimise l'espérance définit par :

$$Q(\theta|\theta^m) = \sum_{i=1}^{n} \sum_{k=1}^{K} ln\ (p_k \cdot \mathcal{N}_k(x_i)) \cdot P(z_i = k|x_i, \theta^m) \tag{3.14}$$

À partir de paramètres initiaux θ^0, l'algorithme itère entre deux étapes : l'étape E, où les termes de $Q(\theta|\theta^m)$ ne dépendant pas de θ sont calculés, et l'étape M, où l'on choisit les paramètres θ^{m+1} tel que $\theta^{m+1} = arg\ max_\theta\ Q(\theta|\theta^m)$.

Plus simplement, nous définissons ces deux étapes à l'itération m par :

- Étape Estimation : estimation des probabilités d'affectation à priori des x_i aux composantes de paramètres θ^{m-1}. Les termes t_{ik} sont re-calculés avec

$$t_{ik} = \frac{p_k \cdot \mathcal{N}(x_i|\theta_k)}{\sum_{j=1}^{K} p_k \cdot \mathcal{N}(x_i|\theta_k)};$$

- Étape Maximisation : estimation des paramètres θ^m à partir des probabilités d'affectation calculées à l'étapes E.

L'étape E correspond à l'estimation des probabilités *à posteriori* $P(x_i|\theta)$ à partir des paramètres θ^{m-1}. Cela revient à calculer la matrice t d'affectation des données aux composantes.

L'étape de maximisation consiste à ré-estimer les paramètres θ^m à partir de la matrice t. Il faut noter que cette phase dépend de la paramétrisation du modèle de mélange : le choix des contraintes sur les proportions de mélange p_k ou sur les matrices de variance Σ_k. Celeux et Govaert [29] ont détaillé la phase de maximisation pour les 14 modèles possibles suivant ces contraintes. Dans le cas des matrices de variance pleines et de proportions de mélange libres, l'estimation des paramètres peut se résumer aux calculs suivants :

- les centres : $\mu_k = \frac{\sum_{i=1}^{n} t_{ik} x_i}{\sum_{i=1}^{n} t_{ik}}$;

- les proportions de mélange : $p_k = \frac{\sum_{i=1}^{n} t_{ik}}{n}$;

- les matrices de variance de forme $[\lambda_k C_k]$ sont définies par :

$$\Sigma_k = \frac{(x - \mu_k) t_k (x - \mu_k)^t}{\sum_{i=1}^{n} t_{ik}}, \tag{3.15}$$

  où le terme t_k correspond à la colonne k de la matrice t_{ik}.

La propriété de convergence de la vraisemblance a été étudiée dans [124]. Il est montré que sous des conditions générales, la suite de valeurs de vraisemblance converge vers une valeur stationnaire. Cette valeur est néanmoins un minimum local de la vraisemblance. L'optimum obtenu dépend du modèle initial θ^0. Nous verrons dans le chapitre 5 plusieurs techniques permettant de trouver un bon minimum de la vraisemblance. La convergence de la vraisemblance est généralement rapide mais peut aussi être très lente, notamment aux alentours d'un col de vraisemblance. Sa complexité est en $O(k \cdot d^2 \cdot n \cdot E)$, où E est le nombre moyen d'itérations de l'algorithme EM. Elle est donc linéaire en fonction du nombre de données.

L'algorithme EM est donc dépendant de paramètres fixés manuellement : le nombre de composantes et les paramètres initiaux du modèle. L'algorithme suivant propose des solutions pour améliorer ces différents points.

3.4.1.3 L'algorithme Stochastique EM (SEM)

L'algorithme SEM de Celeux et Diebolt [25] a pour objectif d'améliorer les faiblesses de l'algorithme EM. L'algorithme est similaire à la méthode EM mais incorpore une nouvelle étape entre les étapes E et M. Cette étape se base sur un principe d'affectation aléatoire : les observations sont affectées aléatoirement aux composantes selon les probabilités t_{ik}. Cet algorithme peut s'interpréter comme une version stochastique de l'algorithme Classification EM (voir section 3.4.2.2).

Comme pour la méthode EM, on part de paramètres initiaux θ^0. Le nombre de composantes est fixé à une valeur K supérieure au nombre réel de composantes. L'itération m est définie par :

- Étape Estimation : estimation des probabilités d'affectation t des x_i aux composantes de paramètres θ^{m-1} ;

- Étape Stochastique : tirage pour chaque x_i de la variable $z_i^m = (z_{ik}^m, k = 1, \ldots, K)$ selon une loi multinomiale d'ordre un et de paramètre t_{ik}. Les termes z_i^m fournissent ainsi une partition des données. Si il existe une composante k telle que la cardinalité de l'ensemble $n_k = \{x_i | z_{ik}^m = k\} \leq d + 1$, l'algorithme est ré-initialisé avec un modèle à $K - 1$ composantes ;

- Étape Maximisation : estimation des paramètres θ^m à partir des probabilités d'affectation t calculées à l'étapes E.

L'algorithme s'arrête après convergence de la vraisemblance. Il permet de résoudre plusieurs défauts liés à l'algorithme EM. Il améliore la convergence aux cols de vraisemblance, minimise l'importance des paramètres initiaux, et permet de déterminer le nombre de composantes. Ce dernier point est réalisé à l'étape S où une composante est supprimée si le nombre d'observations associées à cette dernière

est inférieur ou égale à la dimension des observations. On évite ainsi le problème des matrices de variance singulières (problème explicité dans la suite, section 3.5). D'autres critères peuvent être utilisés, comme par exemple, une valeur minimale de la proportion de mélange. Cette technique pour trouver le bon nombre de composantes n'est pas applicable dans notre contexte. Nous devons pouvoir obtenir des composantes associées à une seule donnée, entraînant une proportion de mélange faible. Ainsi, les deux critères proposés ici ne fonctionnent pas.

L'algorithme n'est pas automatique puisque l'utilisateur doit fournir un nombre initial de composantes supérieur ou égale au vrai nombre de classes dans le modèle. La convergence de la vraisemblance θ est étudiée dans [26]. La suite (θ^m) ne converge pas ponctuellement à cause de l'étape S, où les observations sont affectées aléatoirement. Cet algorithme est moins efficace que l'algorithme EM dans le cas où nous disposons de petits échantillons : l'influence des pertubations aléatoires de l'étape S est trop forte et entraîne une sous-estimation du nombre de composantes. Par contre, le modèle obtenu ne dépend pas de l'état initial : grâce à l'étape S, θ^m ne stagne pas au col de vraisemblance et la convergence est ainsi accélérée.

3.4.1.4 L'algorithme Simulated Annealing EM (SAEM)

L'algorithme SAEM [27] se situe entre les algorithmes EM et SEM. Il présente les mêmes avantages que SEM par rapport à EM, mais améliore la robustesse face aux petits échantillons. Une nouvelle étape est ajoutée entre les étapes S et M. Celle-ci permet de réduire les perturbations aléatoires de l'étape S grâce à une suite de réel positif (γ^m) qui décroît vers 0 lorsque le nombre d'itérations tend vers l'infini, en partant de $(\gamma^0) = 1$. Elle joue le même rôle que la température dans les algorithmes de recuit simulé.

L'algorithme est initialisé avec des paramètres initiaux θ^0 et un nombre de composantes K supérieur au nombre réel de composantes. Les étapes à l'itération m sont :

- Étape Estimation : estimation des probabilités d'affectation t des x_i aux composantes de paramètres θ^{m-1} ;
- Étape Stochastique : tirage pour chaque x_i de la variable $z_i^m = (z_{ik}^m, k = 1, \ldots, K)$ selon une loi multinomiale d'ordre un et de paramètre t_{ik}. Les termes z_i^m fournissent ainsi une partition des données. Si il existe une composante k telle que la cardinalité de l'ensemble $n_k = \{x_i | z_{ik}^m = k\} \leq d + 1$, l'algorithme est ré-initialisé avec un modèle à $K - 1$ composantes ;
- Étape Annealing : calcul de la partition $t_{ik}^{m'} = t_{ik}^m + \gamma^{(m-1)}(z_{ik}^m - t_{ik}^m)$. Une nouvelle température $\gamma^{(m-1)}$ est calculée ;
- Étape Maximisation : estimation des paramètres θ^m à partir des probabilités d'affectation t calculées à l'étape E.

Cet algorithme est itéré jusqu'à convergence du critère de vraisemblance. Son principe est de commencer comme l'algorithme SAEM et d'aboutir à un algorithme EM déterministe. Il converge presque sûrement vers un maximum local de la fonction de vraisemblance, dans le cas où les composantes ont des densités de probabilités appartenant à la famille exponentielle. Mais le comportement théorique et pratique dépend fortement de la vitesse de convergence de la suite (γ^m) vers 0 et de sa décroissance régulière.

Comme pour l'algorithme précédent, la méthode pour déterminer le bon nombre de composantes

n'est pas pertinente pour notre cas d'utilisation. La robustesse de cet algorithme face aux petits échantillons est aussi un point négatif au vu de notre objectif. Enfin, cet algorithme ajoute un nouveau paramètre critique à régler avec la température γ.

3.4.2 Critère du maximum de vraisemblance classifiante et algorithmes pour l'optimiser

Les algorithmes précédents se basent sur l'optimisation de la vraisemblance pour déterminer les paramètres du modèle. Ceux présentés dans cette partie permettent d'estimer simultanément les paramètres du modèle et la partition des observations (les labels z_i).

3.4.2.1 Définition du maximum de la vraisemblance classifiante

La partition z des observations est inconnue. Les couples (x_i, z_i) avec $z_i \in 1 \dots K$ sont ainsi supposés être générés par une loi de distribution de densité $f(x, k|\theta)$. La densité jointe $f(X, z|\theta)$ des couples (x_i, z_i) est le produit de la densité de chaque couple. Elle est appelée *vraisemblance classifiante* de $(\theta|z)$. Son logarithme népérien est défini par :

$$CL(\theta, z) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \cdot ln(p_k \mathcal{N}(x_i|\theta_k)) \tag{3.16}$$

Contrairement au maximum de vraisemblance, l'estimateur du maximum de vraisemblance est inconsistant. Lorsque la taille de l'échantillon tend vers l'infini, l'estimateur ne tend pas systématiquement vers les vrais paramètres θ.

La maximisation de la vraisemblance classifiante consiste donc à déterminer simultanément les paramètres inconnus θ et les labels z_i. Cette méthode est équivalente à optimiser des critères classiques, basés sur des notions géométriques et non liés à la notion de vraisemblance ou encore à une hypothèse de modèle de mélange. Des exemples sont les critères d'inertie intra-classe ou des k-means. Il est montré dans [28] que les partitions obtenues par minimisation de ce dernier critère et par maximisation de la vraisemblance classifiante sont identiques pour le modèle gaussien $[p\lambda I]$.

Nous présentons deux algorithmes permettant de maximiser la vraisemblance classifiante. Selon [11], ces algorithmes sont censés être plus adaptés pour une démarche de classification.

3.4.2.2 L'algorithme Classification EM (CEM)

L'algorithme CEM est une version classifiante de l'algorithme EM [28]. Une étape, entre E et M, construit une partition des observations utilisée pour la ré-estimation des paramètres.

L'algorithme est initialisé avec des paramètres initiaux θ^0 ou une partition initiale z^0. Les étapes à l'itération m sont :

- Étape Estimation : estimation des probabilités d'affectation t des x_i aux composantes de paramètres θ^{m-1} ;
- Étape Classifiante : estimation d'une partition z^m à partir des termes t_{ik} et du critère MAP ;

- Étape Maximisation : estimation des paramètres θ^m à partir de la partition z^m calculée à l'étape C.

Cet algorithme est itéré jusqu'à convergence du critère de vraisemblance classifiante. Comme les algorithmes précédents, il converge vers un optimum local qui dépend souvent fortement des paramètres initiaux. Il est montré dans [23] qu'il peut produire des estimateurs biaisés des paramètres du mélange dans le cas où les composantes sont proches et les proportions de mélanges très différentes. Dans ce dernier cas, l'algorithme CEM manque de robustesse face aux faibles échantillons. Ces défauts sont dus à la nouvelle étape C de l'algorithme. Cet algorithme est donc efficace quand les classes sont distinctes et si les proportions de mélange sont similaires. Il présente par contre l'avantage sur l'algorithme EM d'être plus rapide du point de vue de la convergence.

Dans notre contexte, les classes peuvent être très proches et les proportions de mélanges très différentes, ce qui posent un problème pour cet algorithme. Nos expériences ont montré une surparamétrisation des modèles obtenus sur des données temporelles. Nous n'utiliserons donc pas cet algorithme.

3.4.2.3 L'algorithme Classification Annealing EM (CAEM)

L'algorithme CAEM [28] est une version de type recuit simulé de l'algorithme CEM. Il permet de maximiser la vraisemblance classifiante et de limiter la dépendance aux paramètres initiaux. L'approche est similaire à l'algorithme SAEM : une nouvelle étape permet d'affecter aléatoirement les observations aux composantes. Ces associations sont calculées à partir d'une probabilité d'affectation des observations aux composantes, basée sur une séquence de températures réelles positives $(\gamma^m, m \geq 0)$ décroissant vers 0 quand m tend vers l'infini à partir de $\gamma^0 = 1$.

L'algorithme est initialisé avec des paramètres initiaux θ^0 ou une partition initiale z^0. Les étapes à l'itération m sont :

- Étape Estimation : estimation des probabilités d'affectation t_{ik}^m des x_i aux composantes de paramètres θ^{m-1} ;

- Étape Annealing : calcul des quantités $r_{ik}^m = \frac{(t_{ik}^m)^{(\frac{1}{\gamma^{m-1}})}}{\sum_{k=1}^K (t_{ik}^m)^{\frac{1}{\gamma^{m-1}}}}$. Une nouvelle température γ^m est calculée ;

- Étape Classifiante : calcul pour chaque x_i de la variable $z_i^m = (z_{ik}^m, k = 1, \ldots, K)$ selon une loi multinomiale d'ordre un et de paramètres $(r_{ik}^m, k = 1, \ldots, K)$. Une partition z^m est ainsi définie à partir des variables z_i^m ;

- Étape Maximisation : estimation des paramètres θ^m à partir de la partition z^m calculée à l'étape C.

Cet algorithme est arrêté quand la convergence du critère de vraisemblance classifiante est atteinte. Il consiste à itérer l'algorithme SEM et à tendre vers l'algorithme CEM quand le nombre d'itérations tend vers l'infini. Le choix de la suite γ^m est important pour un bon comportement de l'algorithme et il est recommandé d'avoir une vitesse de convergence assez lente.

La convergence de cet algorithme vers un minimum local de la vraisemblance classifiante ne semble pour le moment pas encore prouvée. Ce procédé semble néanmoins éviter les solutions sous-optimales

que subit CEM. En revanche, il a le défaut de nécessiter un grand nombre d'itérations pour obtenir un bon résultat et il est recommandé de ne l'utiliser que sur des petits échantillons de données. De plus, il présente l'inconvénient d'ajouter le paramètre γ.

Discussion

Tous les algorithmes présentés dans cette section sont dépendant de paramètres fixés manuellement, l'utilisateur devant fournir un nombre de composantes ou une limite supérieure du nombre de classes. Pour cette dernière catégorie, nous avons vu que les critères pour trouver le bon nombre de classes n'étaient pas pertinents. L'utilisation des ces algorithmes telle quelle ne convient donc pas pour notre objectif.

Parmi ces algorithmes, notre préférence va à l'algorithme EM. Il présente de bonnes propriétés de convergence et ne dépend pas d'un paramètre critique comme la température. Le problème de convergence lente aux alentours d'un col de vraisemblance sera limité par notre approche incrémental, où l'on se base sur les paramètres à $t-1$ pour obtenir ceux du nouveau modèle, en prenant en compte la nouvelle donnée. Nous proposerons dans le chapitre 5 un algorithme incrémental basé sur cet algorithme.

3.5 Gestion des petits échantillons de données

Les petits échantillons de données peuvent poser problème lors de la phase d'estimation des paramètres du modèle, en particulier pour les matrices de variance. L'estimation des paramètres μ_k et Σ_k n'est optimale que quand le nombre d'observations tend vers l'infini. En présence de petits échantillons de données, l'estimation de ces paramètres est instables. L'estimation des valeurs propres des matrices de variance est biaisée : les plus grandes valeurs propres sont sur-estimées et les valeurs les plus faibles sont sous-estimées. De plus, quand le nombre d'observations associées à une composante est inférieur à $d+1$, la vraisemblance tend vers l'infini : la matrice de variance devient singulière (non inversible). Comme nous l'avons vu précédemment, les algorithmes d'estimation des paramètres requièrent l'inversion de la matrice de variance. Par exemple pour $d=1$ et un échantillon de 1 observation, le terme $(x-\mu_k)$ de l'équation 3.15, permettant de calculer la matrice de variance d'une composante, est nul puisque $x=\mu_k$. La matrice est ainsi non inversible, et la densité de la donnée x tend vers l'infini, le dénominateur du terme de gauche de l'équation 3.4 étant nul.

Pour répondre aux problèmes de singularité et d'instabilité, des techniques de régularisation des matrices de variance existent. Les trois méthodes de régularisation les plus employées sont la règle discriminante linéaire, l'analyse discriminante régularisée ([47]) et l'estimateur de variance LOOC (*Leave One Out Covariance* [61]). Nous présentons aussi une méthode basée sur une approche bayésienne.

Ces méthodes tentent de réduire l'instabilité des estimations des paramètres obtenues avec de faibles échantillons en cherchant des paramètres plus plausibles. Nous présentons dans cette partie les différentes techniques de régularisation et proposons une technique appropriée à notre problème. Notons que ce problème ne se pose que si le volume des matrices de variance des composantes est libre.

3.5.1 Règle discriminante linéaire

Cette méthode consiste à remplacer les estimations de matrices de variance par une combinaison linéaire de l'ensemble des matrices de variance :

$$\Sigma_p = \frac{(n_1-1)\Sigma_1 + (n_2-1)\Sigma_2 + \cdots + (n_K-1)\Sigma_K}{N-K}, \tag{3.17}$$

où $n = n_1 + \cdots + n_k$ et n_i représente les données associées à la composante i (obtenue par le critère MAP par exemple).

La matrice régularisée Σ_p est une moyenne des matrices de variance, pondérée par les données affectées à chaque composante. Théoriquement Σ_p est un vrai estimateur de Σ_i si nous posons comme hypothèse à priori sur les matrices de variance que $\Sigma_1 = \cdots = \Sigma_K$.

Des expériences [71, 120] ont montré qu'elle peut améliorer la vraisemblance des observations pour des petits échantillons même quand les matrices de variance sont libres. Par contre quand le nombre d'observations devient grand, les résultats obtenus sont moins bons. Le choix entre les matrices de variance estimées Σ_k et Σ_p représente un ensemble limité d'estimation des vraies matrices Σ_k. Cette méthode présente néanmoins l'avantage d'être simple.

Les méthodes présentées dans la suite peuvent être considérées comme des méthodes intermédiaires entre l'estimation Σ_k et Σ_p.

3.5.2 Analyse discriminante régularisée (RDA)

L'analyse discriminante régularisée est une méthode d'optimisation à 2 paramètres permettant de combiner plusieurs formes de matrices de variance : la variance estimée Σ_k, une variance commune et une variance diagonale où les élément diagonaux sont la moyenne des éléments diagonaux de soit la variance commune, soit la variance estimée. La matrice de variance régularisée est définie par :

$$\begin{aligned} \Sigma_k^{rda}(\lambda,\gamma) &= (1-\gamma)\Sigma_k^{rda}(\lambda) + \gamma\Big(\frac{tr(\Sigma_k^{rda}(\lambda))}{d}\Big)I, \\ \Sigma_k^{rda}(\lambda) &= \frac{(1-\lambda)(n_i-1)\Sigma_k + \lambda(n-K)\Sigma_p}{(1-\lambda)n_i + \lambda n}, \end{aligned} \tag{3.18}$$

où $\lambda, \gamma \in [0,1]$, $tr(\Sigma)$ est la trace de la matrice Σ et Σ_p est défini par (3.17).

Le paramètre λ permet de régler la régularisation entre la variance commune et la variance estimée, tandis que le paramètre γ règle le poids de la régularisation avec une variance diagonale. Puisque le terme $tr(\Sigma_k^{rda}(\lambda))/d$ est juste la moyenne des valeurs propres de la variance $\Sigma_k^{rda}(\lambda)$, le paramètre γ a l'effet de réduire les valeurs propres les plus grandes et d'augmenter les plus faibles.

Cette méthode permet ainsi plusieurs alternatives de régularisation. En fixant γ à zéro et en faisant varier λ, nous obtenons une combinaison linéaire entre la règle discriminante linéaire et la variance estimée. En fixant λ à zéro et en faisant varier γ, cela revient à régulariser avec une combinaison linéaire entre une variance diagonale et la variance estimée.

Les paramètres λ et γ sont estimés à partir d'une fonction d'erreur basée sur la classification réelle des observations et d'un ensemble d'apprentissage. Pour chaque observation x_i de l'ensemble d'apprentissage, on itère les étape suivantes :

- enlever l'observation x_i du modèle ;
- calculer tous les centres et toutes les matrices de variance ;
- affecter de façon binaire l'observation x_i (par exemple avec le critère MAP).

La classe de l'observation x_i étant connue, nous déterminons les paramètres λ et γ en minimisant le nombre des mauvaises affectations de toutes les observations de l'ensemble d'apprentissage.

Un désavantage de cette technique est sa complexité. L'estimation des paramètres λ et γ nécessitent de rechercher plusieurs couples de valeurs sur l'intervalle $([0,1] \times [0,1])$ et pour chacun de ces couples, de recalculer les matrices de variance des k classes. La complexité de l'estimation peut être élevée si le nombre de composantes et le nombre de possibilités testées du couple (λ, γ) sont grands : pour chaque couple (λ, γ), il est nécessaire de ré-évaluer toutes les classes. De plus, ces paramètres sont communs à toutes les classes puisque cette technique calcule simultanément toutes les matrices de variance, alors qu'ils ne sont pas obligatoirement optimaux pour toutes les classes. Enfin, il faut noter que la valeur optimale du couple (λ, γ) n'est pas unique.

3.5.3 L'estimation Leave One Out Covariance (LOOC)

Cette méthode de régularisation dépend seulement des variances estimées Σ_k (contrairement à RDA, la matrice moyenne ne dépend plus de poids basés sur le nombre de données associées à chaque composante). Chaque variance est optimisée indépendamment et les paramètres à estimer sont calculés pour chaque composante avec le critère de vraisemblance comme critère d'erreur. La régularisation consiste à combiner les matrices estimées avec une matrice diagonale ou une moyenne :

$$\Sigma_k^{looc}(\tau_k) = \begin{cases} (1-\tau_k)diag(\Sigma_k) + \tau_k\Sigma_k & 0 \leq \tau_k \leq 1 \\ (2-\tau_k)\Sigma_k + (\tau_k - 1)\Sigma_{moy} & 1 < \tau_k \leq 2 \\ (3-\tau_k)\Sigma_{moy} + (\tau_k - 2)diag(\Sigma_{moy}) & 2 < \tau_k \leq 3, \end{cases}$$

où $\Sigma_{moy} = \frac{1}{K}\sum_{k=1}^{K}\Sigma_k$.

Le paramètre τ_k détermine quelle estimation est choisie pour la composante k : si $\tau_k = 0$ alors la matrice de variance diagonale est utilisée ; si $\tau_k = 1$ la matrice estimée est retournée ; et si $\tau_k = 3$, c'est la matrice diagonale de la moyenne des variances estimées qui est considérée. Les autres valeurs de τ_k permettent un mélange entre ces trois possibilités.

Par rapport à la technique RDA, cette approche permet une combinaison en plus : la combinaison entre la variance moyenne et la matrice diagonale obtenue à partir de Σ_{moy} ($2 < \tau_k \leq 3$).

La méthode d'estimation du paramètre τ_k est presque identique à la méthode précédente. Chaque variance Σ_k est estimée séparément en se basant sur son ensemble d'observations associées. Pour chaque observation soustraite, on ré-estime le centre et la matrice de variance de la composante et on re-calcule la vraisemblance des n_k données associées à la composante k. Plusieurs valeurs de τ_k doivent être testées pour chaque composante. Le paramètre optimisant la vraisemblance des observations est sauvegardé. Cette approche permet de se passer de l'ensemble d'apprentissage utilisé précédemment avec la technique RDA. Notons que cette technique peut aussi être utilisée pour estimer les paramètres (λ, γ) de cette dernière technique dans le cas où ces deux paramètres sont différents pour chaque composante.

La méthode LOOC présente donc aussi une complexité élevée mais est néanmoins moins coûteuse que la méthode précédente puisqu'un seul paramètre doit être évalué (contrairement à RDA où deux paramètres doivent être estimés).

Plusieurs autres méthodes similaires, basées sur des combinaisons linéaires de matrices de variance, sont proposées dans [54, 56, 107].

3.5.4 Régularisation bayésienne

Cette méthode, présentée dans les travaux [45, 86], consiste à fixer une contrainte à priori à l'aide d'une loi de probabilité sur les paramètres du modèle. Son principal avantage est de pouvoir limiter

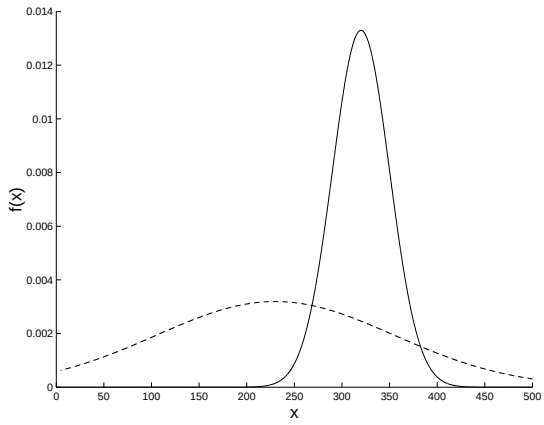


Figure 3.5: Illustration de l'inférence bayésienne pour les paramètres θ : la probabilité a priori (ligne en pointillés) reflète nos connaissances initiales sur les paramètres avant d'observer les données. Une fois les données observées, nous pouvons calculer les probabilités a posteriori (ligne pleine) correspondantes à l'aide du théorème de Bayes. Certaines valeurs des paramètres étant plus consistantes que d'autres avec les paramètres, la densité obtenue est plus fine que la distribution a priori.

l'impact de la régularisation en fonction de la taille de l'échantillon disponible : plus sa taille augmente et plus la matrice régularisée doit tendre vers la matrice estimée. Le poids de la régularisation diminue en fonction de l'information disponible sur les vraies paramètres du modèle.

L'approche consiste à décrire l'incertitude des valeurs des paramètres à l'aide d'une densité de probabilité *à priori*. Après avoir observé des données, le théorème de Bayes permet de trouver les densités de probabilités *à posteriori* :

$$P(\theta|X) = \frac{P(X|\theta) \cdot P(\theta)}{P(X)} \tag{3.19}$$

La connaissance des données entraîne une densité *à posteriori* plus élevée pour les paramètres représentant l'ensemble X. La figure 3.5 ci-après illustre l'inférence bayésienne pour les paramètres θ.

Des contraintes à priori peuvent être définies sur chaque paramètres du modèle (les centres, les proportions de mélanges et les matrices de variance). Pour une question de simplicité de calcul, ces lois *a priori* sont des lois *a priori conjuguées* (une loi *a priori* telle que sa forme fonctionnelle de sa probabilité *a posteriori* soit identique). Les lois utilisées sont généralement :

- la loi normale pour les centres ;
- la loi de Dirichlet pour les proportions de mélanges ;
- la loi Γ (en dimension 1) et la loi de Wishart (en dimension > 1) pour les matrices de variances.

Les paramètres des lois *a priori* sont à définir manuellement : dans le cas des matrices de variances, ils ont une influence sur le volume et la forme (seulement en dimension 2) des composantes. Poser une contrainte sur le volume maximum d'une composante est dans notre contexte délicat puisque l'étalement des classes n'est pas connu.

Nous prenons l'exemple concret de l'estimation d'une variance. La figure 3.6 ci-après présente un

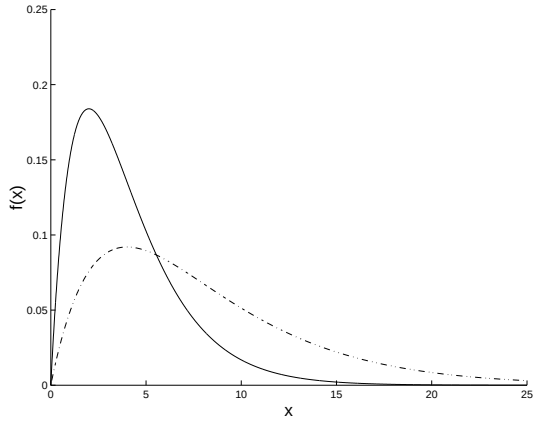


Figure 3.6: Exemple de densité Gamma pour différentes valeurs de paramètres : $\alpha = 2$, $\beta = 2$ *(ligne pleine) et* $\alpha = 2$, $\beta = 4$ *(ligne en pointillés).*

exemple de densités de la loi Γ obtenues pour plusieurs valeurs des paramètres α et β. Les courbes fixent un à priori sur la taille maximum des composantes : pour la courbe en ligne pleine, la connaissance à priori fixe la variance des composantes avec une valeur proche de 4.

L'estimation des paramètres est obtenue en optimisant le maximum *a posteriori* des données, et non la vraisemblance. Cela revient à optimiser le critère :

$$L_p(\theta) = L(\theta) + lnP(\theta), \tag{3.20}$$

où $P(\theta)$ est une probabilité *a priori* sur les paramètres du modèle. Pratiquement cette méthode de régularisation s'intègre directement dans l'algorithme EM. L'étape E de l'algorithme EM reste identique et l'étape M, pour la mise à jour de la matrice de variance, devient :

$$\Sigma'_k = \frac{\sum_{i=1}^{n} p(x_i|\theta_k)(x_i - \mu'_k)(x_i - \mu'_k)^t + 2\boldsymbol{\beta}}{\sum_{i=1}^{n} p(x_i|\theta_k) + 1 + 2(\boldsymbol{\alpha} - (d+1)/2)},$$

où α et β sont les paramètres de la loi Γ.

La régularisation avec cette méthode est plus stable que les méthodes précédentes : la régularisation est réalisée conjointement avec l'optimisation du maximum à posteriori. Notons que cette méthode et les méthodes de régularisation linéaire peuvent être utilisées simultanément [108, 111].

3.5.5 Proposition d'une régularisation

Notre régularisation des matrices de variance doit permettre d'obtenir des composantes avec un volume libre tout évitant les composantes ayant une forme très aplaties, expliquée par un rapport entre les valeurs propres des matrices élevé (par exemple la variance de la composante de la figure 3.7(a), page 55).

Les méthodes de régularisation basées sur une combinaison linéaire de l'ensemble des matrices de variance présentent le problème suivant : ils posent l'hypothèse qu'il existe une dépendance entre les groupes de données à classer. La régularisation d'une composante est obtenue à partir d'une combinaison linéaire des autres composantes. Un petit échantillon ne doit pas être associé à une composante ayant un

grand volume sous prétexte que les autres variances présentent cette propriété. L'approche bayésienne présente quant à elle l'inconvénient de fixer un volume maximum au matrice de variance, ce qui est peu pertinent dans notre contexte puisque l'étalement des classes n'est pas connu. Nous proposons donc notre propre technique de régularisation.

Dans le cas temporel, où la dimension des données est de 1, nous nous limitons à fixer un volume minimum à chaque matrice de variance quand celle-ci est singulière :

$$\Sigma'_k = \Sigma_k + \varphi \tag{3.21}$$

Pour le cas spatial, notre solution consiste à estimer les matrices de variance en imposant deux contraintes :

- un volume minimum est imposé, en fixant des valeurs propres des matrices de variance minimales. Cela permet de gérer le cas où la composante est associée à une seule donnée ;

- un ratio maximum entre les deux valeurs propres de la matrice de variance est fixé. Une matrice pleine est plus flexible qu'une matrice diagonale puisque l'orientation des axes est libre. Mais, avec seulement deux observations, les deux valeurs propres ont des magnitudes différentes et les observations ont des densités de vraisemblance excessives. Cette technique permet d'éviter d'obtenir de telles composantes.

Soit la matrice de covariance Σ_k :

$$\Sigma_k = \lambda_k \mathbf{D}_k \mathbf{A}_k D'_k, A = \begin{bmatrix} a_{1k} & 0 \\ 0 & a_{2k} \end{bmatrix}, \tag{3.22}$$

où $a_{1k} > a_{2k}$.

Les contraintes imposées ci-dessus sont donc appliquées en régularisant la matrice de covariance avec :

$$\Sigma_k = U. \begin{bmatrix} max\ (\beta, a_{1k}) & 0 \\ 0 & max\ (\varphi, a_{2k}, \varphi' \cdot a_{1k}) \end{bmatrix} .U^{-1}, \tag{3.23}$$

où φ et φ' sont deux paramètres fixés manuellement. $\varphi' \in [0, 1]$ fixe le ratio maximal entre les deux valeurs propres et φ est la valeur minimale des valeurs propres.

La figure 3.7 ci-dessus présente un exemple de modèle obtenu avec notre régularisation des matrices de covariance. La matrice de covariance régularisée de la composante présente un plus grand volume que la composante. La valeur propre minimale a_{2k} de la composante k non-régularisée a été augmentée puisque $a_{2k} < \varphi \cdot a_{1k}$. Cette technique permet de gérer le cas des petits échantillons et permet de défavoriser les composantes de formes aplaties. En effet, la log-vraisemblance négative obtenue pour le modèle régularisé de la figure 3.7 est 12.19 alors qu'elle est de 8.39 pour le modèle non-régularisé. Ce critère devant être minimisé, la régularisation a bien permis de donner moins de poids aux vraisemblances des observations. Cet aspect permet ainsi d'améliorer le choix du modèle quand nous comparerons des modèles avec un nombre différent de composantes. Dans cet exemple, un modèle à deux composantes serait plus approprié.

Dans le cas où une composante est associée à une seule donnée, sa vraisemblance tend vers l'infini. Cela peut poser problème quand nous comparerons différentes hypothèses de modèles (problème traité dans le chapitre suivant), en favorisant excessivement les classes à une seule donnée. L'utilisation de la log-vraisemblance et la pondération par la proportion de mélange de la densité permettent de limiter

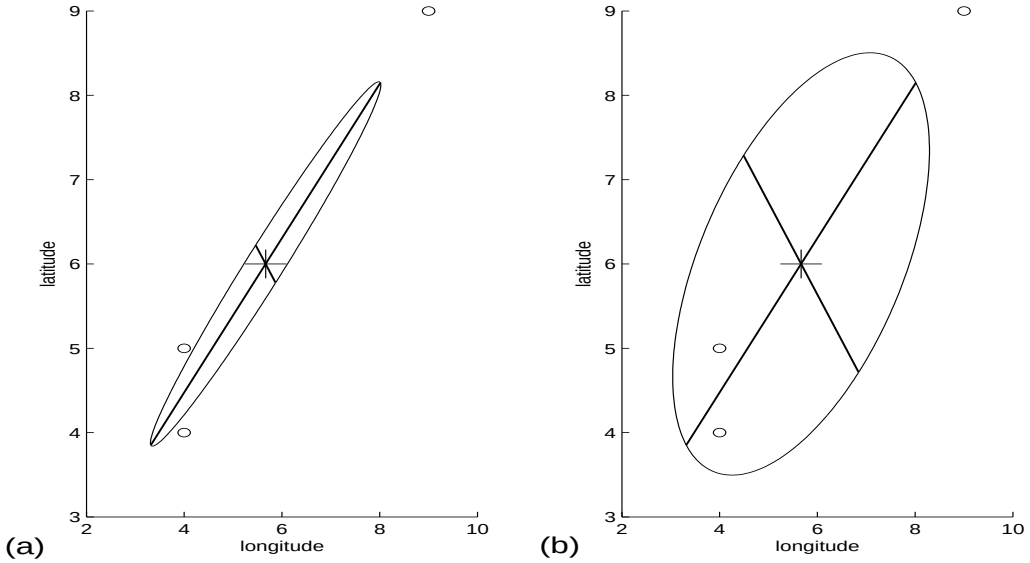


Figure 3.7: Comparaison des matrices de covariance non-régularisées et régularisées : la figure (a) présente le modèle obtenu avec une matrice non-régularisée tandis que sur la figure (b), nous avons appliqué notre régularisation. "+" représente le centre de la composante et l'ellipse sa covariance. Le jeu de données est identique dans les 2 *cas.*

la valeur de vraisemblance d'une telle donnée par rapport aux autres. Cette technique permet ainsi de trouver un bon compromis entre une composante associée à une seule donnée et un modèle où cette donnée est associée à une composante adjacente.

Notre approche présente l'avantage d'être simple et de faible complexité. Les seuls paramètres critiques à régler sont la taille minimum des composantes et le ratio entre la valeurs propres.

3.6 Conclusion

Nous avons présenté dans ce chapitre la modélisation de nos données à l'aide d'un modèle de mélange gaussien. Les deux avantages de ce modèle pour notre cas d'utilisation sont :

1. les algorithmes d'estimation des paramètres présentent une complexité raisonnable (linéaire en fonction du nombre de données), un point pertinent pour que notre application soit implémentable sur un terminal mobile ;

2. Les composantes gaussiennes sont sensibles aux données isolées, un cas de figure que l'on retrouve régulièrement dans les métadonnées spatiales et temporelles d'une collection d'images personnelles et que l'on veut mettre en valeur dans les partitions à obtenir.

Deux approches distinctes pour estimer les paramètres d'un modèle de mélange ont été présentées, basées sur la vraisemblance et la vraisemblance classifiante. Parmi les algorithmes détaillés, l'algorithme

EM présente les meilleures propriétés pour concevoir un algorithme incrémental :

- il n'est pas robuste face aux petits échantillons, contrairement aux algorithmes SEM et SAEM ;
- il présente de bonnes propriétés de convergence avec tous types de données et toutes contraintes à priori.

Le problème de convergence lente aux alentours d'un col de vraisemblance sera limité par notre approche incrémentale.

Enfin, la gestion des petits échantillons posant un problème d'estimation des paramètres du modèle, nous avons proposé une technique de régularisation des matrices de variance. Cette technique consiste à poser des contraintes sur la forme et le volume minimum des composantes. La technique est simple à mettre en oeuvre et n'augmente pas la complexité de l'algorithme d'estimation des paramètres.

La modélisation de nos données étant définie, un problème reste à résoudre : déterminer la complexité d'un modèle de mélange. Dans le chapitre suivant, nous étudions différents critères statistiques permettant de comparer des modèles de mélange gaussien.

CHAPITRE 4

Sélection de la complexité dans le cadre de modèles de mélange

Nous présentons dans ce chapitre les différentes techniques pour déterminer la complexité d'un modèle de mélange. Notre choix se porte sur l'approche par pénalisation de la vraisemblance. Nous détaillons les différents critères numériques existants et motivons ensuite notre préférence pour le critère ICL (*Integrated Completed Likelihood*) par des résultats expérimentaux.

4.1 Introduction

Nous avons présenté dans le chapitre précédent notre modélisation des métadonnées spatiales et temporelles. Étant dans un contexte de classification, le nombre de composantes dans un modèle n'est pas connu : il est nécessaire de le déterminer afin de pouvoir fournir un système automatique. Une méthode doit être proposée pour comparer plusieurs solutions de modèles dans un espace d'hypothèses des paramètres et sélectionner celui représentant au mieux la structure des données.

Plusieurs techniques existent pour déterminer le meilleur modèle. Nous présentons dans ce chapitre plusieurs méthodes, dans le cadre des modèles en général : la méthode par ré-échantillonnage, la pénalisation de la mesure de qualité d'une partition par sa complexité. Ensuite nous présentons plusieurs méthodes liées aux modèle de mélange. Parmi ces solutions, nous verrons que la plus appropriée est la sélection à l'aide de critères numériques pénalisant la vraisemblance par la complexité ou la classifiabilité du modèle.

Nous proposons ensuite de présenter plusieurs types de critères numériques permettant de déterminer la complexité d'un modèle de mélange. On peut distinguer deux types de critères basés sur deux approches différentes : l'approche mélange et l'approche par classifiabilité. La première approche est basée sur des critères statistiques appelés *critères d'information*. Certains sont établis dans le contexte de la théorie bayésienne (BIC, AWE) et d'autres s'appuient sur des justifications de statistique classique (AIC, $AIC3$, $ICOMP$). Ces critères sont pertinents pour déterminer les paramètres (contraintes sur les proportions et les matrices de variances) mais aussi le nombre de composantes pour certains. Les critères de classifiabilité sont eux exclusivement utilisés pour obtenir le nombre de composantes dans un modèle. Nous présentons les critères PC, NEC, E et LP. Enfin ces différentes approches peuvent être combinées ensemble : les derniers critères proposés (C, LC, ICL) pénalisent la vraisemblance par un critère de classifiabilité et par la complexité du modèle.

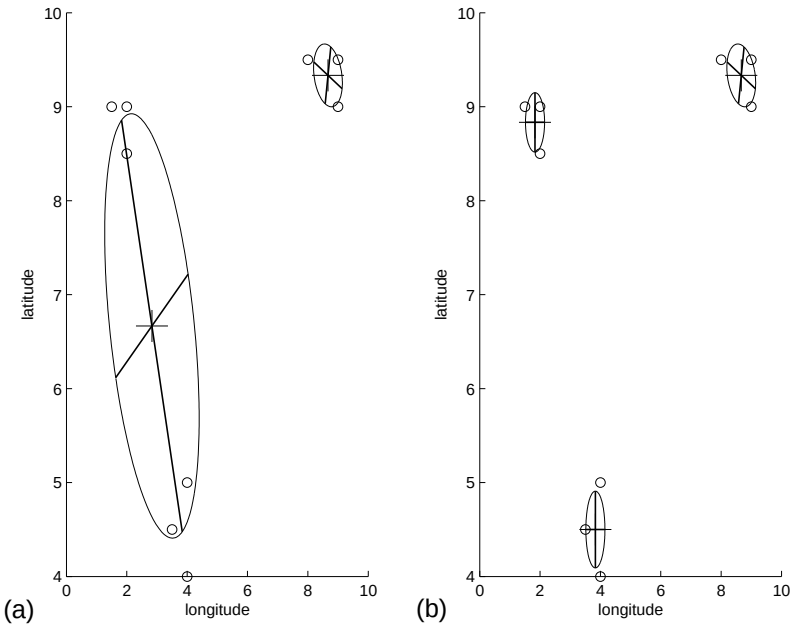


Figure 4.1: comparaison de deux modèles avec des hypothèses différentes. Le modèle sur la figure (a) a 2 *composantes tandis que celui sur la figure (b) en contient* 3*. Le critère de sélection doit favoriser ici le deuxième modèle puisqu'il semble le plus approprié au vu des observations*

4.2 Méthodes de sélection d'un modèle

Notre premier objectif est de choisir une méthode pour sélectionner le modèle représentant au mieux la structure réelle des données. La figure 4.1 ci-dessus présente deux modèles de mélange obtenus sur un même jeu de données. La technique doit ainsi nous permettre de sélectionner le modèle (b), composé de trois composantes, qui semble être le plus pertinent pour représenter nos données.

Nous présentons dans cette partie plusieurs approches pour la sélection de modèles. Pour chacune d'entres elles nous évaluons leur pertinence pour fournir un algorithme incrémental.

4.2.1 Méthode par ré-échantillonnage

Plusieurs travaux [67, 75, 105] posent l'hypothèse que la structure des données peut être retrouvée à partir d'un sous-ensemble des données. Ils proposent de déterminer la complexité d'un modèle de mélange en comparant les estimations des paramètres sur plusieurs échantillons de données obtenus à partir de l'ensemble initial.

Cette méthode par ré-échantillonnage est liée à la simulation statistique, dite de Monte Carlo [70]. Elle repose sur des techniques de génération d'échantillons artificiels à partir d'un seul échantillon de données et est utilisée lorsque la complexité analytique du problème ne permet pas d'inférence classique. Elle consiste à répéter des analyses sur différents échantillons de données simulés puis à étudier les fluctuations des résultats. La différence fondamentale entre la méthode de Monte Carlo et celle par ré-échantillonnage est que dans la première les données sont artificielles, tandis que pour la deuxième les

simulations sont basées sur des données réelles. Plusieurs méthodes de validation permettent de créer de différentes manières les échantillons de données : la permutation [43], la validation croisée [66], le Jackknife [91] et le Bootstrap [40].

La première méthode consiste à créer un échantillon à partir d'une permutation des données initiales. La validation croisée divise l'ensemble des données initiales en plusieurs sous-ensembles de taille variable. Le Jackknife propose de comparer plusieurs sous-ensembles de données obtenus en enlevant une seule donnée de l'ensemble initial (cette technique est généralement utilisée pour détecter les données isolées). Enfin le Bootstrap consiste à simuler m échantillons de même taille n que l'échantillon initial. Ils sont obtenus par tirage au hasard avec remise parmi l'ensemble initial (certaines données auront un poids élevé et d'autres un poids nul).

La méthode par ré-échantillonnage propose de comparer l'estimation des paramètres du modèle de mélange sur plusieurs échantillons de données obtenus avec une des méthodes précédentes. Il est nécessaire de définir une fonction d'erreur pour comparer les différents tests. La complexité du modèle optimisant ce critère permet ensuite d'obtenir le modèle pertinent pour représenter la structure des données. Nous présentons ci-dessous plusieurs exemples de travaux déterminant la complexité d'un modèle de mélange à partir de la méthode par ré-échantillonnage.

Application pour la sélection de modèles de mélange

Les travaux de Levine et al. [67], Roth et al. [96] et Smyth [105] ont une approche basée sur la validation croisée. Ils consistent à diviser les données en ensembles d'apprentissage et de validation. Les paramètres des modèles sont estimés à l'aide des ensembles d'apprentissage et sont ensuite évalués sur les ensembles de validation. Le modèle le plus stable sur les ensembles de validation, selon un critère d'erreur à définir (la vraisemblance par exemple pour les travaux de Smyth [105]), est ainsi retenu.

L'approche par permutation est un peu différente des autres, comme le prouvent les travaux de Vasko et al. [116]. Cette proposition fonctionne avec des séries temporelles. Elle consiste à estimer les paramètres sur les observations initiales et à évaluer le modèle obtenu sur une permutation de cet ensemble. La permutation des observations permet de créer une série temporelle aléatoire présentant une distribution identique aux données initiales. L'ensemble obtenu permet de représenter le bruit contenu dans les observations. Chaque modèle est évalué à l'aide d'un critère d'erreur et les variations d'erreurs entre les modèles obtenus pour l'ensemble des observations initiales et l'ensemble permuté sont comparées. Le modèle retenu est celui présentant une variation d'erreur similaire sur les deux ensembles de données.

4.2.2 Pénalisation de la mesure de qualité d'un modèle

Tout modèle dispose d'un critère pour évaluer sa qualité : par exemple le critère d'erreur quadratique pour un algorithme hiérarchique ou la vraisemblance pour un modèle de mélange. L'approche consiste à trouver un compromis entre un tel critère et la simplicité de la partition. Nous présentons deux méthodes, la pénalisation par la complexité et la détection d'un coude dans la mesure de qualité. Cette dernière méthode est en fait similaire à la première.

4.2.2.1 Pénalisation par la complexité

Cette approche est basée sur le principe d'Occam qui consiste à sélectionner le modèle le plus simple pour bien décrire les données. Il s'agit de trouver un compromis entre la qualité du modèle et sa simplicité. Si on se base seulement sur un critère de qualité (ou d'erreur) de la partition, plus le modèle sera

complexe et plus nous obtiendrons un modèle s'adaptant parfaitement à la structure des données, mais présentant peu d'intérêt. Le critère de vraisemblance ou le critère d'erreur quadratique sont optimisés pour un nombre de composantes croissant. Un modèle comprenant n observations présentera une mesure de qualité maximale pour un modèle de n classes : chaque classe est associée à une observation. Un tel résultat n'est bien sûr pas pertinent puisque la classification obtenue n'a plus aucun intérêt, notre objectif étant de pouvoir résumer les observations. L'objectif de cette approche est de pénaliser le critère de qualité par la complexité du modèle (le nombre de paramètres libres).

Dans le contexte de modèles de mélange, plusieurs critères numériques ont été proposés dans la littérature, pénalisant la vraisemblance par le nombre de paramètres libres du modèle : les critères AIC (*Akaike Information Criterion* [2, 3]), AIC3 et ICOMP (*Informational Complexity Criterion*) [20, 21, 22]. Nous reviendrons sur ces critères par la suite. Cette approche présente une similarité avec l'approche bayésienne (voir section 4.2.3.2).

Pratiquement, pour comparer deux modèles, il suffit de calculer le critère numérique pour chacun d'entre eux et de sélectionner ensuite le modèle l'optimisant. Cette technique permet d'évaluer des modèles de complexités différentes à l'aide d'une fonction, généralement rapidement et facilement calculable. La comparaison entre deux modèles est ainsi possible d'une manière directe. Par contre pour obtenir le "meilleur" modèle, il est toujours nécessaire de tester un ensemble de modèles entre 1 et K composantes.

4.2.2.2 Méthode par coude d'erreur

Cette approche consiste à détecter dans une courbe d'erreur, décroissante au fur et à mesure que la complexité augmente, un changement significatif de pente. La complexité du modèle obtenu au moment du "coude" de la courbe est censée fournir le modèle représentant au mieux la structure des données. Cette approche n'a pas de fondement théorique mais semble plutôt empirique : l'idée est d'augmenter la complexité du modèle et de détecter quand la variation de l'erreur en fonction de la complexité devient faible. Elle revient à sélectionner, comme la méthode précédente, un compromis entre la complexité et la mesure de qualité.

Il est nécessaire de disposer d'une courbe avec en abscisse le nombre de composantes dans le modèle et en ordonnée la valeur de l'erreur pour chaque modèle testé. L'algorithme de classification pour obtenir les paramètres des modèles est libre mais il est préférable d'utiliser un algorithme hiérarchique (agglomératif ou par division). Cela évite d'avoir à recalculer l'ensemble des paramètres pour passer du modèle de k composantes à celui à $k \pm 1$ (puisque dans le cas des algorithmes hiérarchiques, on se base sur les paramètres du modèle obtenu à l'itération $i - 1$). La fonction d'erreur étant libre, nous pouvons choisir n'importe quelles évaluations métriques. L'approche hiérarchique est aussi préférable pour obtenir le critère d'erreur d'un modèle puisqu'il peut s'obtenir à partir du modèle précédent (en calculant la variation d'erreur entraînée par la fusion ou la division de composantes). Un exemple de critère d'erreur peut être tout simplement la vraisemblance des données [11] ou la distance de Kullback-Leibler (voir l'exemple ci-après).

Une fois la courbe d'erreur obtenue, l'objectif est ensuite de détecter un coude indiquant un changement de variation significatif. La pente de la courbe d'erreur est censée être fortement décroissante pour les premiers modèles de faible complexité, ceci s'expliquant par une grande différence entre les modèles. Pour les modèles trop complexes, la variation d'erreur devient beaucoup plus faible puisque les modèles sont de plus en plus similaires. Le changement de pente dans la courbe fait apparaître un coude dans la courbe, lié au modèle censé présenter la meilleure complexité pour représenter la structure des données.

La figure 4.2 ci-dessus présente un exemple de courbe d'erreur obtenue avec un algorithme hiérar-

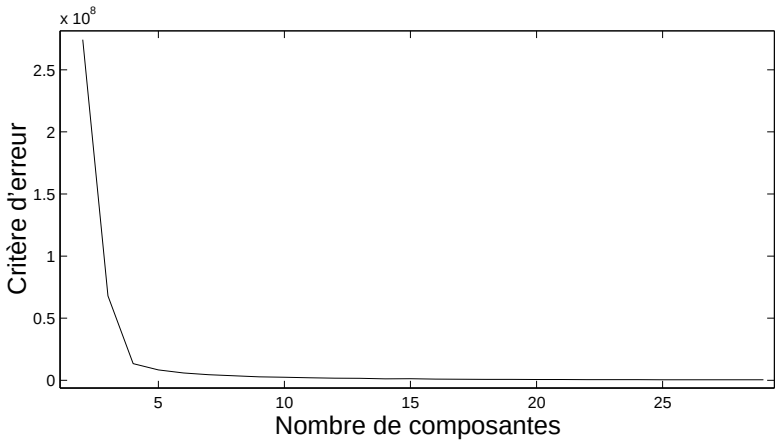


Figure 4.2: Détection d'un coude dans une courbe d'erreur. En abscisse le nombre de composantes dans le modèle et en ordonnée la valeur de l'erreur pour chaque modèle testé. Pour le modèle à 4 *composantes, nous observons un coude dans la courbe : ce modèle est censé être une bonne solution pour représenter les données.*

chique. Le critère d'erreur est celui proposé par Goldberger et Roweis [53], basé sur une divergence modifiée de Kullback-Leibler. La distance entre les modèles deux à deux est en abscisse et la valeur du critère d'erreur est en ordonnée. La courbe est divisée en trois parties : la partie droite, où la courbe est quasiment linéaire et de faible pente, indique que les modèles sont très similaires ; la partie gauche présente une pente abrupte et signifie que les modèles sont différents et peu homogènes entre eux ; et enfin le coude qui représente un compromis entre les deux extrêmes. Choisir un modèle dans la région du coude de la courbe fournira un nombre raisonnable de composantes. Dans cet exemple, le modèle représentant bien la structure des données est composé de 4 classes. Le coude dans la courbe est bien obtenu pour le modèle contenant 4 composantes.

Plusieurs techniques existent pour déterminer ce coude :

- la plus grande amplitude entre deux points ;
- le premier point avec une dérivée seconde inférieure à une limite statique ;
- le point avec la plus grande dérivée seconde ;
- le point le plus loin d'une droite obtenue à partir de la courbe.

D'autres méthodes plus complexes existent, par exemple les travaux de Salvador [97], cherchant deux droites linéaires pour résumer les deux parties significatives de la courbe d'erreur et choisissant leur intersection pour déterminer le nombre de composantes dans le modèle.

Des méthodes similaires à la recherche d'un coude existent [112, 113], en transformant la courbe d'erreur en une nouvelle fonction, et en sélectionnant le minimum ou le maximum pour déterminer le modèle adéquat.

4.2.3 Approches liées aux modèles de mélange

Nous présentons dans cette section des méthodes utilisées avec les modèles de mélange : les tests d'hypothèses, les tests d'hypothèses bayésiens et la pénalisation de la vraisemblance par un critère de classifiabilité.

4.2.3.1 Tests d'hypothèses

Cette méthode consiste à tester une hypothèse nulle H_0 (celle à laquelle on croit le plus) contre une hypothèse alternative H_1 (le *challenger*), avec $H_0 : \theta \in \Theta_0$ et $H_1 : \theta \in \Theta_1$. H_0 et H_1 correspondent respectivement à deux modèles M_O et M_1. Le rapport des vraisemblances maximales est défini par :

$$r(X) = \frac{max_{\theta \in \Theta_0} f(X|\theta)}{max_{\theta \in \Theta_0 \cup \Theta_1} f(X|\theta)} \tag{4.1}$$

Dans le cas où $r(X)$ est de petite valeur, l'hypothèse H_0 est rejeté, et si $r(X)$ est proche de 1, H_0 est accepté. Le problème est le calcul de ce rapport de vraisemblance : il est souvent impossible à obtenir sauf dans le cas asymptotique en supposant certaines conditions de régularité. Cela pose aussi des problèmes pour déterminer la complexité d'un modèle. De plus, un défaut de ce test est qu'il accepte facilement H_0 pour des petits échantillons de données (et réciproquement H_1 pour un grand échantillon).

4.2.3.2 Approche Bayésienne

L'approche bayésienne consiste à retenir le modèle le plus probable à posteriori. Le calcul de cette probabilité nécessite d'intégrer la vraisemblance sur l'espace Θ des paramètres. Ce calcul étant difficile pour une grande dimension de Θ, il est plus fiable d'approximer le rapport entre deux vraisemblances intégrées, défini par le facteur de Bayes. Une approximation de ce facteur est fournie par les critères numériques BIC et AWE. Ces deux critères ont une forme similaire aux critères pénalisant la vraisemblance par la complexité du modèle mais découlent d'une explication théorique différente. Nous reviendrons sur ces critères dans la suite.

4.2.3.3 Pénalisation par la classifiabilité

Le contexte probabiliste fournit une autre manière de pénaliser la vraisemblance des données : l'utilisation d'un critère de classifiabilité. De tels critères ont pour objectif de retenir le modèle avec des classes bien séparées et des données bien regroupées. Ils se basent sur des informations sur la classification, obtenues à posteriori. Dans le contexte des modèles de mélange, des exemples de critères numériques sont : l'entropie de la matrice d'affectation modèle-données ou le critère PC, basés tous les deux sur les probabilités d'appartenance *a posteriori* des données aux classes. Comme nous le verrons ces critères peuvent aussi être utilisés seuls.

Discussion

Nous avons besoin d'une méthode présentant une complexité en temps de calcul faible afin de nous permettre de comparer rapidement différentes solutions de modèle : en particulier, pouvoir comparer un modèle obtenu au temps $t-1$ avec plusieurs solutions au temps t. Nous devons pouvoir comparer les modèles deux à deux.

La méthode par ré-échantillonnage présente un inconvénient pour notre approche. Il est nécessaire d'avoir à disposition l'ensemble des données à classer, ce qui n'est pas possible dans notre cas puisque

les données ne sont connues qu'au fur et à mesure. La détection par coude d'erreur présente quant à elle l'inconvénient de devoir obtenir une courbe d'erreur pour plusieurs complexités : on ne peut pas comparer directement les modèles deux à deux. Dans l'optique d'un système incrémental, il serait nécessaire pour chaque nouvelle donnée de recalculer le critère d'erreur pour toutes les complexités (entre 1 et K) et de re-détecter le coude dans la courbe. Cette approche ne nous semble donc pas pertinente pour concevoir notre système. Enfin le test d'hypothèses présente une trop grande difficulté de calcul.

Notre choix se porte donc sur la méthode par pénalisation de la vraisemblance défini à partir de l'une des trois approches décrites précédemment : l'approche bayésienne et la pénalisation de la vraisemblance par la complexité ou la classifiabilité. Ces critères sont en général faciles à obtenir et présentent l'avantage de pouvoir directement comparer les modèles deux à deux. Nous détaillons dans la suite plusieurs critères numériques et leurs propriétés : tout d'abord les critères pénalisant la vraisemblance par la complexité, et ensuite ceux la pénalisant par la classifiabilité.

4.2.4 Les critères numériques basés sur une pénalisation de la vraisemblance par la complexité

Nous présentons dans cette section les différents critères numériques existants qui découlent de la théorie bayésienne et d'approches statistiques plus classiques.

4.2.4.1 Approche bayésienne

Cette approche consiste à retenir le modèle M le plus probable à posteriori. Cette probabilité s'exprime par :

$$P(M|X) \propto P(X|M)P(M) \tag{4.2}$$

Dans le cas où tous les modèles sont à priori équiprobables, cela revient à sélectionner le modèle optimisant $P(X|M)$. Cette probabilité, appelée *vraisemblance intégrée* ou *vraisemblance marginale*, est définie par :

$$p(X|M) = \int_{\Theta} \mathcal{N}(X|\theta, M)P(\theta|M)d\theta$$

où $\mathcal{N}(X|\theta, M)$ est la densité de mélange du modèle M et $P(\theta|M)$ la loi *a priori* de θ sous ce modèle. Pour calculer cette probabilité, il est nécessaire de définir une loi *a priori* sur θ et d'évaluer l'intégrale. L'évaluation analytique étant difficilement réalisable, diverses approximations existent [65] : les méthodes numériques, les méthodes de Monte-Carlo, ou l'approximation de Laplace-Metropolis (voir [84] pour les deux dernières). Une alternative consiste à approcher la log-vraisemblance marginale par le maximum de log-vraisemblance pénalisé par une fonction du nombre de paramètres libres et du nombre de données dans l'ensemble X. L'expression obtenue est plus simple et permet de s'affranchir de définir une loi *a priori*. Cette approximation est basée sur le facteur de Bayes, qui permet d'estimer le rapport entre deux vraisemblances intégrées.

En notant M_0 un modèle en compétition avec le modèle M, on peut exprimer la probabilité *a posteriori* $P(M|X)$ par :

$$\begin{aligned} P(M|X) &= \frac{B_0 P(M)}{\sum_{m' \in H} B'_0 P(M')} \\ &\propto B_0 P(M) \end{aligned}$$

où $B_0 = \frac{P(X|M)}{P(X|M_0)}$. Le terme B_0, appelé facteur de Bayes, est le rapport entre les vraisemblances intégrées des deux modèles M et M_0. Si on ne pose aucun à priori sur les modèles et que $\forall M, M'$ sur l'ensemble d'hypothèses H, $P(M) = P(M')$ alors on obtient :

$$\begin{aligned} P(M|X) &\propto B_0 \\ &\propto P(X|M) \end{aligned}$$

Il est plus robuste d'estimer le rapport des vraisemblances intégrées du facteur de Bayes (sans estimer aucune vraisemblance intégrée) que d'estimer la vraisemblance intégrée. Le facteur de Bayes permet d'introduire les critères BIC et AWE, détaillés dans la suite.

Les critères BIC [100] (*Bayes Information Criterion*) et AWE [7] (*Approximate Weight of Evidence*) sont deux approximations du facteur de Bayes B_0. Le facteur de Bayes est obtenu à partir du critère de Schwarz [100], permettant de calculer une approximation de $2lnB_0$:

$$S = 2L(M) - 2L(M_0) - (q(M) - q(M_0))ln(n),$$

où $L(M)$ est le maximum de log-vraisemblance avec le modèle M, $q(M)$ le nombre de paramètres libres dans ce modèle. Le terme S est généralement appelé le critère (BIC).

Si on doit comparer plus de deux modèles, il suffit de calculer le facteur de Bayes entre chaque modèle et le modèle de référence M_0. Le modèle retenu est celui qui maximise le facteur de Bayes. Cela revient à choisir le modèle qui minimise le critère BIC :

$$BIC(M) = -2L(M) + q(M)ln(n) \tag{4.3}$$

Le critère AWE se base sur une approximation différente du facteur de Bayes. Contrairement au critère BIC, ce critère suppose les contraintes à priori sur les paramètres du modèle gaussien connues et se limite aux choix du nombre de composantes. En imposant que les proportions des classes soient égales, l'estimation asymptotique devient :

$$-2lnB_0 \approx -2CL(k) + CL(1) + 2(q(k) - q(1))(\frac{3}{2} + ln(n)),$$

où B_0 est le facteur de Bayes de k classes contre un modèle M_0 à une seule classe et $CL(k)$ le maximum de log-vraisemblance classifiante pour k classes. Comme pour le critère BIC, la constante due au modèle M_0 peut être abandonnée. On obtient le critère AWE :

$$AWE(k) = -2CL(k) + 2q(k)(\frac{3}{2} + ln(n)) \tag{4.4}$$

Le modèle retenu est celui qui minimise ce terme.

Enfin, l'approche bayésienne présente un lien avec la théorie du codage MDL. Un critère similaire au critère BIC est le critère MDL (*Minimum Length Description*) [94] :

$$MDL(M) = -L(M) + q(M)ln(n) \tag{4.5}$$

Ce critère présente une pénalisation avec le nombre de paramètres libres plus faible que pour le critère BIC.

Il faut noter que ces critères ne sont censés être efficaces que pour des échantillons de données de grande taille. Néanmoins nous verrons lors des tests expérimentaux que même avec de petits échantillons de données, les résultats sont corrects.

4.2.4.2 Pénalisation de la vraisemblance par la complexité du modèle

Les critères précédents reviennent à pénaliser la vraisemblance par la complexité du modèle et le nombre de données. Nous présentons dans cette section une généralisation de ce type de critère.

La vraisemblance peut être interprétée comme une mesure de qualité de l'ajustement de la densité aux données. Néanmoins ce critère présente le désavantage de toujours choisir le modèle le plus complexe. Des critères pénalisant la vraisemblance par la complexité du modèle ont ainsi été proposés pour régler le problème. Ils ont pour objectif de trouver un compromis entre la vraisemblance d'un modèle et sa complexité. Leur définition générale est :

$$\textit{critère}(M) = -2L(M) + g(M), \tag{4.6}$$

où $g(m)$ est une fonction permettant de calculer la complexité d'un modèle $M \in H$. On sélectionne le modèle qui minimise ce critère.

La complexité d'un modèle dépend en général du nombre de paramètres libres qu'il contient. La fonction g est ainsi de la forme

$$g(m) = \tau q(m), \tag{4.7}$$

où $\tau \in \mathbb{R}$ et $q(m)$ est le nombre de paramètres libres dans le modèle M. Le paramètre τ permet donc de régler le compromis entre la vraisemblance et la complexité du modèle. Plus ce paramètre est élevé et plus le modèle obtenu présentera une complexité faible. Le problème est de déterminer la valeur de τ. Notons que sa valeur optimale peut ne pas exister puisque la complexité n'est peut être pas une fonction linéaire du nombre de paramètres.

Les critères bayésiens BIC et AWE explicités précédemment ont une forme identique à l'équation (4.6). Néanmoins leur fondement théorique permet de fournir une valeur au paramètre τ. Notons que celui-ci est une fonction de la taille n des observations. Nous présentons maintenant les critères AIC, $AIC3$ et $ICOMP$. Les critères AIC et $AIC3$ sont très utilisés puisqu'ils présentent l'avantage d'être simples à obtenir.

Critère AIC

Le critère AIC (*Akaike Information Criterion*) [2, 3] est historiquement le premier critère de pénalisation de ce genre. Il est défini par :

$$AIC(M) = -2L(M) + 2q(M) \tag{4.8}$$

Le terme τ est égal à 2. Ce choix a été justifié dans [3, 4] par deux raisons. $AIC(M)$ est un estimateur du risque moyen de choisir l'estimateur du maximum de vraisemblance sous le modèle M. Ce risque est calculé en se basant sur la distance de Kullback-Leibler entre la vraie distribution et la distribution avec les paramètres estimés par le maximum de vraisemblance. Les comparaisons de modèles par AIC sont asymptotiquement équivalentes à celles fondées sur les facteurs de Bayes. Mais d'autres recherches prouvent que ce choix pour la valeur du paramètre τ est discutable [68].

Le critère AIC est obtenu en se basant sur la théorie classique du test des hypothèses. Ce critère n'est donc pas théoriquement pertinent pour déterminer le nombre de composantes dans un modèle. Pour résoudre ce problème, une variante de AIC, appelé AIC3 [18, 19], a été proposée avec $\tau = 3$:

$$AIC3(M) = -2L(M) + 3q(M) \tag{4.9}$$

La pénalisation avec les paramètres libres du modèle est ici plus forte que pour le critère AIC.

Critère ICOMP

Le critère $ICOMP$ (*Informational Complexity Criterion*) [20, 21, 22] présente la particularité d'utiliser une mesure de complexité non linéaire. Celle-ci se base sur la matrice d'information de Fisher et le nombre de paramètres du modèle. Ce critère est défini par :

$$ICOMP(M) = -2L(M) + q(M)ln(\frac{trF_N^{-1}(M)}{q(M)}) - ln|F_N^{-1}(M)|, \tag{4.10}$$

où $F_n(M)$ est la matrice d'information de Fisher du modèle M. Le calcul de ce critère est néanmoins beaucoup moins évident que celui des critères précédents : la matrice de Fisher est difficile à obtenir.

Comportement de ces critères

Les expériences proposées dans [33] ont montré que le critère AIC surestime fortement le nombre de composantes K. Le critère $AIC3$ présente le même défaut mais la surestimation est plus faible. Par contre, le critère BIC a tendance à sous-estimer le nombre de composantes. Le critère AWE, quant à lui, le sous-estime fortement. Ce résultat est normal puisque la pénalisation avec les paramètres libres est plus forte que pour le critère BIC. Enfin le critère $ICOMP$ présente de meilleurs résultats que ces trois critères.

4.2.5 Critères en classification automatique

Les critères présentés dans cette section permettent de choisir un modèle en classification automatique. Les critères explicités précédemment sont aussi utilisables pour déterminer un modèle adéquat mais ils ne prennent pas en compte la qualité de la partition obtenue. Deux ensembles de critères sont détaillés ici. Les premiers sont des critères mesurant la classifiabilité des observations : ils favorisent les modèles avec des composantes distinctes. Leur principal défaut est qu'ils ne peuvent pas comparer un modèle à une composante avec un modèle en comprenant plusieurs. Les seconds sont des critères pénalisant la vraisemblance par un critère de classifiabilité. Ils permettent d'éviter le problème de comparaison avec un modèle ayant une seule composante.

4.2.5.1 Critère de classifiabilité

Ces critères permettent de sélectionner des modèles avec des composantes distinctes et des observations bien regroupées. Les composantes obtenues ont des paramètres séparés et les observations sont fortement affectées à une seule composante. Une telle partition est facilement interprétable et est définie par des probabilités *a posteriori* proches de 0 ou de 1.

Cette approche consiste donc à dire qu'une composante ne doit exister que si elle est suffisamment distincte des autres composantes. Dans le cas où deux composantes sont proches, cela pose des problèmes d'interprétation de la partition puisque il est difficile de déterminer à quelle composante les observations sont associées. Il se pose la question de l'utilité d'une des deux composantes, une seule pouvant être plus pertinente pour représenter l'ensemble de leurs données associées.

Nous présentons les critères PC, E, LP, NEC et MIR. Ces critères ont peu de fondement théorique et s'appuient sur des notions de bon sens. Leur principal défaut est qu'ils ne peuvent pas comparer un modèle de 1 composante et un modèle en comprenant 2 et plus. Le modèle à une composante est dit "absorbant", le critère étant optimisé pour ce cas ou incalculable (critère NEC).

Critère PC

Le critère PC (*Partition Coefficient*), proposé dans [10], est défini par :

$$PC(K) = \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik}^2, \tag{4.11}$$

où le terme t_{ik} est la probabilité *a posteriori* que la donnée i soit générée par la composante k (terme de la matrice t), n est le nombre de données et K le nombre de composantes. Ce critère est calculé après l'optimisation de la vraisemblance du modèle. Quand le nombre de composantes est de 1 ce critère est au maximum : on ne peut donc pas comparer la solution à une composante. Enfin ce critère a tendance à sous-estimer le nombre de composantes dans le modèle.

Critère d'entropie E

Le critère d'entropie est défini par :

$$E(K) = -\sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik} \cdot ln(t_{ik}) \tag{4.12}$$

Les probabilités d'affectation des données aux composantes sont obtenues au maximum de vraisemblance. Le modèle choisi est celui minimisant ce critère. Il présente le même défaut que le critère précédent : on ne peut comparer le cas du modèle monoclasse aux autres modèles.

Critère LP

Le critère LP (logarithme de la probabilité de la partition) est similaire au critère d'entropie :

$$LP(K) = -\sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \cdot ln(t_{ik}), \tag{4.13}$$

où t_{ik} est obtenu pour le maximum de vraisemblance et z_{ik} est la partition obtenue avec le critère MAP. Ce critère ne permet pas de comparer le cas monoclasse.

Critère NEC

Le critère NEC (*Normalized Entropy Criterion*) a été proposé par Celeux et Soromenho [30] et est défini par :

$$NEC(K) = \frac{E(K)}{L(K) - L(1)}, \tag{4.14}$$

où $L(K)$ est le maximum de vraisemblance pour K classes et $E(K)$ l'entropie de la matrice d'affectation modèle-données (équation (4.12)). L'entropie fournit une mesure de qualité du modèle et la différence $L(K) - L(1)$ est la qualité des données sous K composantes par rapport à un modèle avec une seule composante. NEC consiste à trouver un compromis entre ces deux termes.

On note que $NEC(1)$ n'est pas calculable puisque $L(1) - L(1) = 0$. Celeux et Soromenho ont néanmoins proposé une procédure empirique pour les mélanges gaussiens multivariés :

- estimer les paramètres du mélange par maximum de vraisemblance pour différentes valeurs de k et choisir k^* minimisant $NEC(k)$;

- pour comparer $k = 1$ et $k = k^*$, estimer les paramètres d'un mélange de k^* classes gaussiennes de centres égaux ($\mu_1 = \cdots = \mu_{k^*} = \sum_{i=1}^{n} \frac{x_i}{n}$) et de proportions égales ($\alpha_1 = \cdots = \alpha_{k^*} = \frac{1}{n}$).

Critère MIR

Enfin, le critère MIR (*Minimum Information Ratio*) est proposé dans [123]. Nous ne le détaillons pas ici, pour plus d'information, il faut se reporter à [11]. Ce critère se base sur les matrices d'information de Fisher qui sont difficiles à obtenir. De nombreuses variantes de MIR existent : $AMIR$ (Adjusted MIR), ANC (Adjusted Number of Components) et WID (Within Component Discrepancy) [33]. Ces critères ont tendance à sous-estimer le nombre de classes et les résultats obtenus semblent peu convaincants [33].

4.2.5.2 Vraisemblance pénalisée par classifiabilité

L'approche mélange et les critères de classifiabilité ne sont pas totalement adaptés à notre objectif de classification. Les critères mélanges ne prennent pas en compte l'aspect classification, et les critères de classifiabilité favorisent excessivement cet aspect au détriment de l'aspect mélange. L'idée proposée dans [11] consiste à pénaliser la vraisemblance avec un critère de classifiabilité afin de trouver un compromis entre les deux approches. Cela est traduit par la pénalisation de la log-vraisemblance par un terme mesurant la classifiabilité des données :

$$\begin{aligned} \textit{critère}(M) &= L(M) - g(M) \\ &= (\textit{Proximité modèle-données}) - (\textit{classifiabilité des données par le modèle}) \end{aligned} \tag{4.15}$$

La définition de ce critère est similaire à l'approche mélange pénalisée par la complexité du modèle. La différence porte sur le terme $g(M)$ qui pénalise la log-vraisemblance avec un critère de classifiabilité.

On retient le modèle optimisant ce critère. Deux stratégies sont possibles pour le calculer :

- stratégie 1 : on maximise la vraisemblance $L(M)$ et ensuite on calcule le terme $g(M)$ à l'aide des paramètres obtenus ;

- stratégie 2 : on maximise le critère globalement.

Ces deux stratégies présentent en pratique peu de différences du point de vue des résultats. Il a été retenu dans [11] de proposer deux critères, C et CL, basés sur le terme d'entropie E et le terme LP.

Critère C

Le critère C pénalise la log-vraisemblance par l'entropie de la matrice d'affectation modèle-données :

$$C(M) = L(M) - E(M), \tag{4.16}$$

où $L(M)$ est la log-vraisemblance optimisée et le terme $E(M)$ est calculé à partir des paramètres obtenus. Ce critère permet de trouver un compromis entre la vraisemblance des données et leur entropie. La vraisemblance est ainsi pénalisée quand les composantes du modèle ont des paramètres proches. Cela

permet de favoriser les modèles avec une bonne séparabilité des classes. Nous notons que la comparaison avec le modèle mono-classe n'est plus absorbant.

Critère CL

Le critère CL pénalise la log-vraisemblance par le critère LP :

$$CL(M) = L(M) - LP(M) \tag{4.17}$$

La stratégie 1 consiste à optimiser la log-vraisemblance et à obtenir les labels z avec le critère MAP. La stratégie 2 consiste quant à elle à obtenir le maximum de vraisemblance classifiante. On note CLM le critère obtenu avec la première stratégie et CL celui obtenu avec la deuxième. Le problème de ce dernier critère est qu'il a tendance à choisir le modèle le plus complexe si les espaces des modèles en compétition sont emboîtés. La conséquence est qu'on ne peut choisir efficacement entre deux nombres de classes dont les modèles gaussiens ont des proportions libres. On peut néanmoins l'utiliser pour sélectionner le nombre de classes si les modèles gaussiens ont des proportions égales. Les résultats expérimentaux [11] ont montré que C sélectionne des modèles plus simples que CLM et que ce dernier choisit des modèles plus simples que CL.

Notons que le critère AWE peut aussi être classé en tant que critère pénalisant la log-vraisemblance par un critère de classifiabilité puisqu'il utilise le maximum de vraisemblance classifiante. Ce critère pénalise la vraisemblance par la complexité du modèle et sa classifiabilité.

4.2.6 Vraisemblance pénalisée par la classifiabilité et la complexité du modèle

Le dernier critère présenté est le critère ICL (*Integrated Completed Likelihood*), proposé dans [12]. Ce critère combine les approches de pénalisation précédentes. La vraisemblance est pénalisée par la complexité du modèle et le critère de classifiabilité E. Il est défini à partir du critère BIC :

$$\begin{aligned} ICL &= -L(M) + \frac{1}{2} \cdot q(M) \cdot ln(n) + E \\ &= \frac{1}{2} \cdot BIC + E, \end{aligned} \tag{4.18}$$

où E est le critère d'entropie (équation (4.12)).

Ce critère pénalise un peu plus la vraisemblance que le critère BIC à l'aide de l'entropie de la matrice d'affectation modèle-données. Des expériences [12] ont montré que ce critère est plus robuste face aux données non-gaussiennes que le critère BIC. Pour ce type de données, le critère BIC a tendance à surestimer la complexité du modèle, les données étant divisées en plusieurs composantes proches. L'entropie permet de pénaliser ce type de configuration.

4.3 Comparaison des critères pour notre cas d'utilisation

Étant dans un processus de classification, nous choisissons les critères pénalisant la vraisemblance par la classifiabilité et la complexité du modèle. On compare dans cette partie les critères suivants : BIC, AWE, MDL, AIC, $AIC3$, C, CLM, ICL. Les critères basés sur la matrice d'information de Fisher

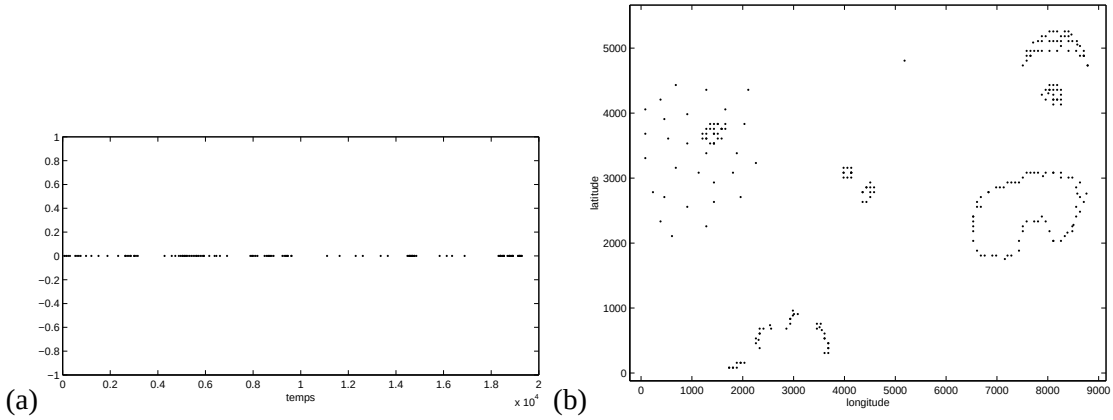


Figure 4.3: Jeux de données temporelles (a) et spatiales (b) utilisés pour tester les différents critères. Les points représentent les données temporelles ou spatiales.

ne sont pas comparés ici. Le calcul de cette matrice est coûteux en temps de calcul. Notre algorithme de classification doit avoir une complexité minimum pour pouvoir fonctionner sur un appareil mobile.

Les expériences proposées consistent à comparer les modèles obtenus avec les différents critères sur deux jeux de données spatiales et temporelles. Chaque critère est optimisé avec une stratégie similaire et nous comparons ensuite les classifications obtenues. Notre objectif est d'obtenir des classes distinctes avec des données bien regroupées.

Les figures 4.3(a) et (b) ci-dessus présentent respectivement les données temporelles et spatiales utilisées pour les expériences de ce chapitre. Elles ont été créées manuellement et présentent des structures variées afin de mettre en valeur les propriétés des différents critères. Les données temporelles contiennent des classes bien regroupées et d'autres plus étalées, et les données spatiales comprennent des données isolées, éparpillées et des formes de classes diverses. Ces jeux de données représentent des cas possibles que l'on pourrait obtenir à partir de métadonnées réelles.

La stratégie d'optimisation de la vraisemblance est la stratégie *em-EM*, présentée dans [13]. D'après les résultats expérimentaux de ces travaux, cette stratégie semble fournir de bons résultats. Le chapitre suivant détaille cette technique et présente d'autres alternatives. Pour un nombre fixe de composantes, cette technique consiste à :

- initialiser aléatoirement les paramètres avec un algorithme de type k-means et appliquer de courtes itérations de l'algorithme EM. Cette étape est répétée plusieurs fois ;

- itérer l'algorithme EM à partir de la position présentant la meilleure optimisation de la vraisemblance obtenue à l'étape précédente.

Cette stratégie est appliquée sur des modèles comprenant entre 1 et K classes et nous retenons le modèle optimisant le critère statistique. Pour les critères pénalisant la vraisemblance par la classifiabilité, nous optons pour la stratégie consistant à optimiser la vraisemblance indépendamment du critère de pénalisation.

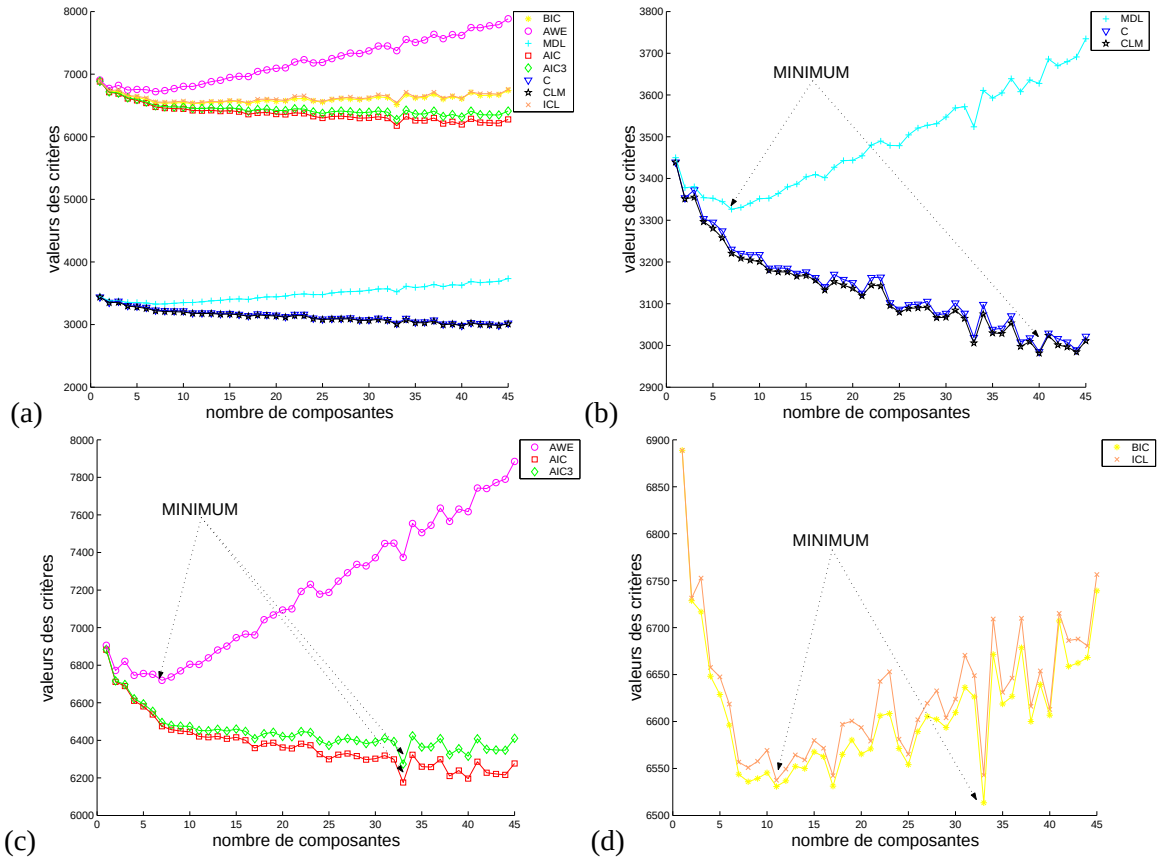


Figure 4.4: Valeurs des différents critères pour la classification temporelle selon le nombre de composantes. L'ensemble des courbes est présenté sur la figure (a), et les figures (b), (c) et (d) détaillent respectivement les critères $\{MDL, C, CLM\}$, $\{AWE, AIC, AIC3\}$ et $\{BIC, ICL\}$.

4.3.1 Classification temporelle

La figure 4.4(a) ci-dessus présente la variation des différents critères pour des modèles comprenant entre 1 et 45 composantes. Nous distinguons deux groupes différents : les critères $\{BIC, AWE, AIC, AIC3, ICL\}$ et $\{MDL, CLM, C\}$. Cela s'explique par le poids de la vraisemblance dans la définition des critères.

Du point de vue de la variation des courbes, nous notons que les critères peuvent être regroupés en quatres classes : $\{C, CLM\}$,$\{AWE, MDL\}$, $\{AIC, AIC3\}$ et $\{BIC, ICL\}$, ce qui correspond aux similarités de leurs définitions. La figure 4.5 ci-après présente les différents modèles obtenus pour chaque critère.

C et CLM

La ressemblance dans les variations des critères C et CLM est cohérente au vu de la similarité entre les deux critères (figure 4.4(b)). Leurs courbes sont toutes les deux décroissantes, ce qui nous paraît normal. Plus le modèle contient de composantes et plus la vraisemblance est améliorée. La pénalisation ne rentre en jeu que lorsque des composantes ont des paramètres proches, entraînant une augmentation

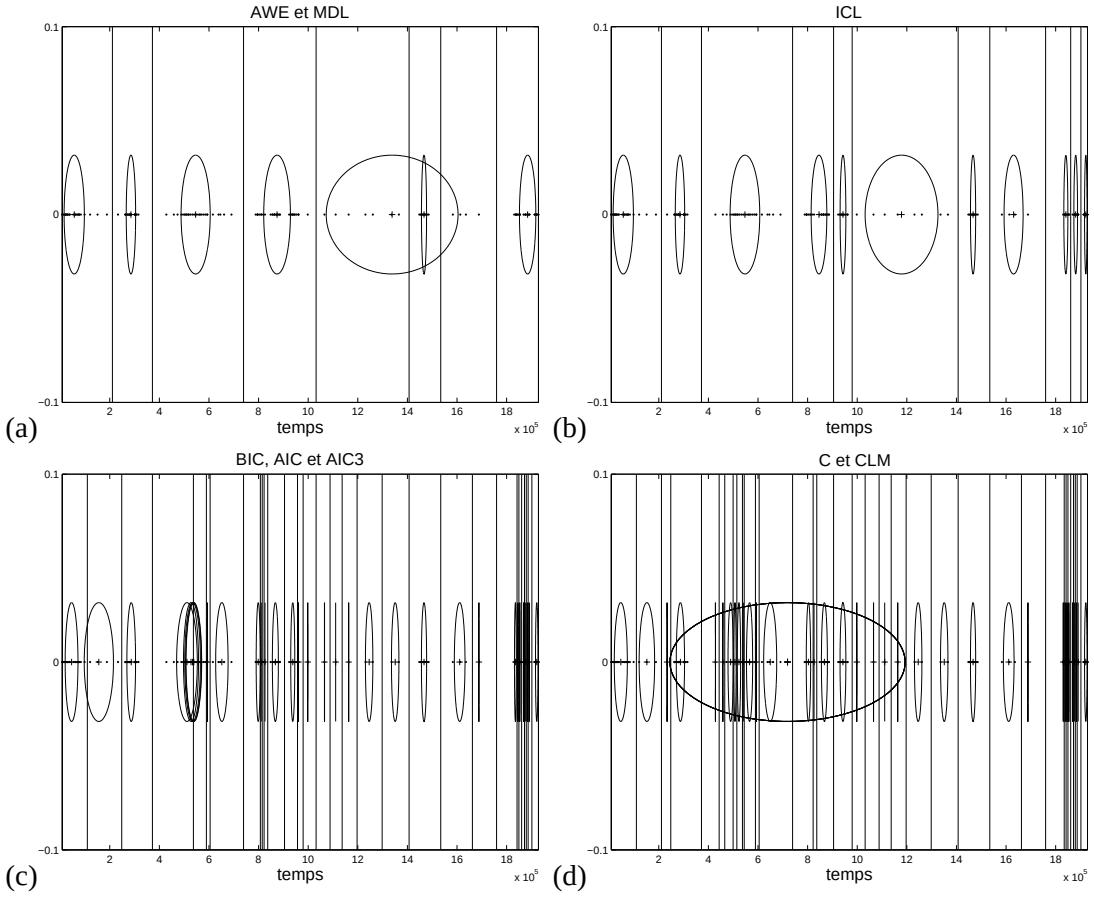


Figure 4.5: Les différentes partitions temporelles obtenues pour chaque critère statistique. Les composantes sont représentées par leur centre (+) et leur variance (ellipse). Les lignes définissent les limites entres les composantes. Elles sont obtenues en parcourant les données selon l'axe temporel et en posant une limite si deux données successives ne sont pas associées à la même composante selon le critère MAP.

de l'entropie (ou de l'entropie classifiante). Le modèle optimum est identique pour les deux critères et contient 40 composantes. Il est présenté par la figure 4.5(d). Nous observons une sur-paramétrisation du modèle, le nombre de composantes étant trop élevé au vu de la structure des données. La composante avec une grande variance, englobant plusieurs autres composantes, n'est associée à aucune donnée selon le critère MAP et semble ici peu pertinente pour notre objectif de classification. Ces critères ne semblent pas assez pénaliser la vraisemblance.

MDL et AWE

Les critères MDL et AWE présentent les courbes avec la plus forte croissance (respectivement figures 4.4(b) et 4.4(c)). La vraisemblance est dans les deux cas fortement pénalisée par le nombre de données et de paramètres libres. Le modèle retenu, présenté par la figure 4.5(a), est composé de 7 classes. La structure des données est assez bien mise en valeur mais elle manque un peu de détail. Ces deux critères nous semblent présenter une pénalisation de la vraisemblance un peu trop élevée.

Nous notons encore la présence d'une classe avec une large variance englobant une autre composante. Ce cas de figure semble plus pertinent que le cas rencontré précédemment dans la figure 4.5(d). Nous obtenons une hiérarchie de composantes : une structure intéressante pour résumer les données.

AIC, AIC3 et BIC

Les courbes des critères AIC et $AIC3$ ont une variation identique (la courbe de $AIC3$ est légèrement au dessus à cause d'une plus forte pénalisation par le nombre de paramètres libres, voir figure 4.4(c)). Le modèle retenu est le même que pour le critère BIC. Il est présenté par la figure 4.5(c) et contient 33 composantes. La classification nous semble ici trop détaillée.

ICL

Le critère ICL présente quelques différences avec le critère BIC, dues à la pénalisation par l'entropie. Leurs courbes respectives sont détaillées dans la figure 4.4(d). Le modèle obtenu avec le critère ICL est présenté par la figure 4.5(b) et est composé de 11 composantes. Ce modèle nous semble le plus approprié pour représenter la structure des données, les limites entre les classes étant visuellement justifiées. La complexité du modèle semble plus cohérente au vu de la structure des données. Le critère ICL semble donc ici le plus approprié pour sélectionner un modèle de mélange gaussien.

4.3.2 Classification spatiale

La figure 4.6 ci-après présente la variation des différents critères statistiques pour un nombre de composantes compris entre 1 et 45. Nous obtenons les mêmes similarités entre les différents critères que pour l'expérience précédente. La figure 4.7 page 76 présente les différentes classifications obtenues pour chaque critère.

AWE et MDL

Les modèles obtenus avec les critères AWE et MDL sont présentés par les figures 4.7(a) et (b) (page 76) et contiennent respectivement de 5 et 6 composantes. Ils sont très similaires, la seule différence étant l'absence, pour le modèle de AWE, de la structure hiérarchique située en $[1000; 3500]$ du modèle de MDL. La composante la plus large, présente dans les deux modèles et englobant la donnée isolée située loin en $[5000; 4500]$, n'est pas très pertinente. Elle suggère que la vraisemblance est ici trop pénalisée pour permettre l'ajout d'une composante pour cette donnée.

AIC, AIC3, C et CLM

Les modèles fournis par les critères $AIC3$ et $\{AIC, C, CLM\}$ sont respectivement présentés par les figures 4.7(d) et (e), page 76. Le modèle de $AIC3$ contient 24 composantes, présentant ainsi trop de détail. Les données éparpillées sur la gauche sont divisées en trois composantes, et le cercle de données sur la droite est divisé en plusieurs petites classes. Le modèle des critères $\{AIC, C, CLM\}$ est encore plus détaillé, celui-ci étant composé de 44 composantes. La classification des données sur la droite présente peu d'intérêt puisque nous obtenons presque une composante par donnée. La composante présentant la variance la plus large n'est associée à aucune donnée (selon le critère MAP) et est donc inutile dans notre contexte de classification. Pour ces trois critères, la vraisemblance n'est pas assez pénalisée.

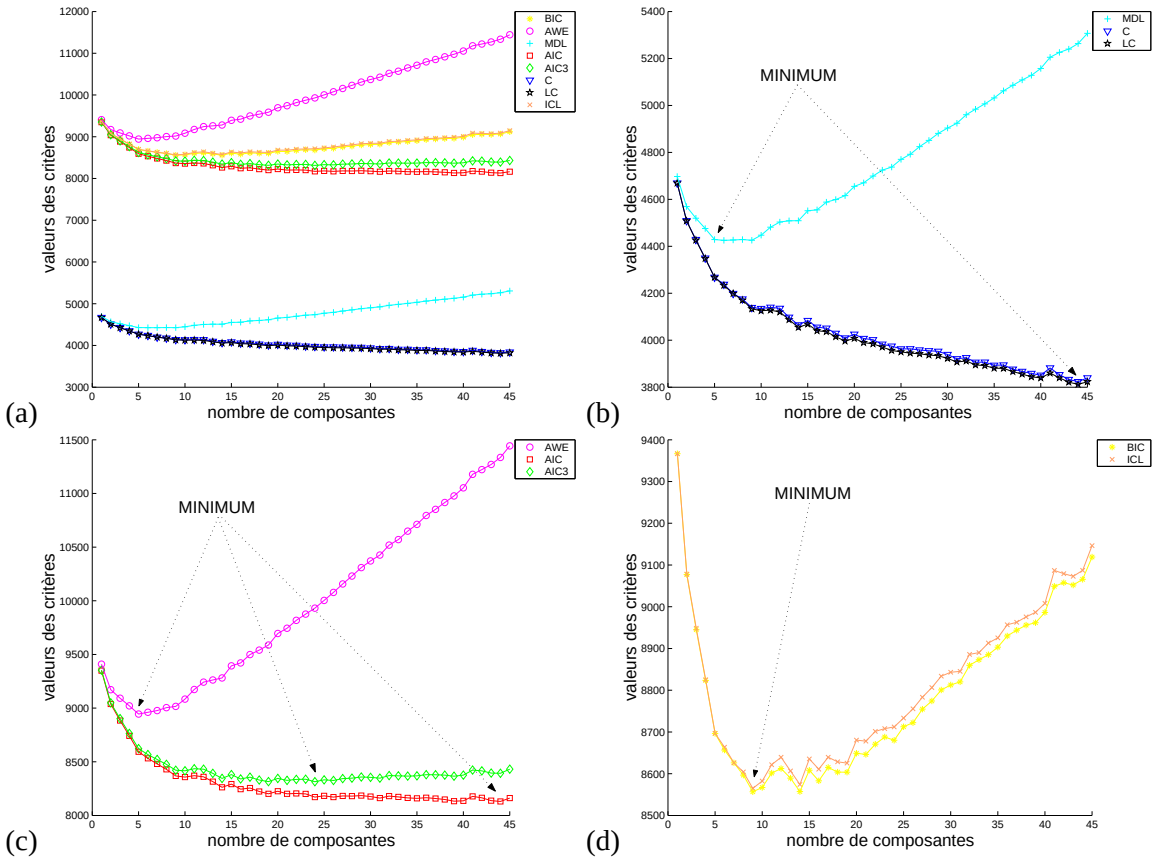


Figure 4.6: Valeurs des différents critères pour la classification spatiale selon le nombre de composantes. L'ensemble des courbes est présenté sur la figure (a), et les figures (b), (c) et (d) détaillent respectivement les critères $\{MDL, C, CLM\}$, $\{AWE, AIC, AIC3\}$ et $\{BIC, ICL\}$.

BIC et ICL

Le modèle le plus pertinent, présenté par la figure 4.7(c), est obtenu avec les critères BIC et ICL. Il contient 9 composantes. Nous observons que la structure hiérarchique sur la gauche, les données en cercles, et la donnée isolée (en $[5000; 4500]$) sont convenablement retrouvées. Ce modèle présente le meilleur compromis entre le jeux de données et le nombre de composantes.

Les critères ICL et BIC semblent donc les plus pertinents pour les métadonnées spatiales. Ils sont très similaires mais comme nous le montrons dans la suite, le critère ICL peut quelquefois présenter un meilleur résultat, notamment dans le cas de groupes de données de formes non-gaussiennes.

La figure 4.8(a) page 77 présente un jeu de données spatiales non-gaussiennes : les données sont issues de tirages aléatoires selon des lois carré uniforme (à gauche) et courbe (à droite). Nous appliquons le même algorithme d'optimisation que précédemment et comparons les résultats pour les critères BIC et ICL pour un nombre de composantes variant entre 1 et 3. La figure 4.8(b) montre la variation des critères BIC et ICL. Le critère BIC présente un minimum pour le modèle à 3 composantes, le groupe en forme de courbe étant divisé en deux composantes, comme le montre la figure 4.8(c). Le critère ICL est minimisé pour le modèle avec deux composantes, présenté par la figure 4.8(d). Dans ce cas de figure, le critère ICL fournit un modèle plus pertinent que le critère BIC. La différence vient de la

pénalisation par l'entropie de la matrice d'affectation modèle-données : pour le modèle à 3 composantes, le chevauchement des deux composantes sur le groupe de données en forme de courbe entraîne une entropie élevée. Pour ce type de données non-gaussiennes, le critère BIC penche nettement en faveur de plus de 2 classes pour trouver une explication "gaussienne par morceau". Il a tendance à fournir des modèles comprenant plusieurs composantes avec des paramètres proches. Le critère ICL peut donc fournir un léger avantage sur le critère BIC, dans le cas où les données ont des formes non-gaussiennes. Dans notre contexte, les métadonnées rencontrées ne présentent généralement pas une configuration gaussienne.

Nous optons donc pour le critère ICL pour sélectionner les modèle de mélange gaussien des classifications temporelle et spatiale. Ce critère semble présenter un bon compromis entre la pénalisation et la complexité du modèle. La pénalisation par l'entropie permet de prendre en compte un aspect classification qui lui donne un avantage sur le critère BIC.

4.4 Conclusion

Parmi les différentes méthodes de sélection de modèles présentées, l'approche par pénalisation de la vraisemblance nous semble la plus appropriée. Son principal avantage est la possibilité de comparer deux à deux les modèles, contrairement aux autres approches qui nécessitent de connaître l'ensemble des données ou de tester un ensemble de modèles pour différentes complexités. Les critères numériques permettent ainsi plus de souplesse pour évaluer les modèles, avec un coût de calcul faible.

Nous avons présenté différents types de critères, que l'on peut regrouper en quatre familles : les critères bayésiens, les critères pénalisant la vraisemblance par la complexité, les critères pénalisant la vraisemblance par un critère de classifiabilité, et enfin les critères pénalisant la vraisemblance par la complexité et la classifiabilité. Les expériences ont montré que le critère ICL, appartenant à la dernière famille, semble le plus approprié pour obtenir un bon modèle, du point de vue classification.

Le critère de sélection de modèles étant choisi, il est nécessaire de disposer d'un algorithme efficace pour rechercher la meilleure optimisation dans l'espace des paramètres. Nous proposons dans le chapitre suivant des méthodes pour déterminer un bon optimum local d'un critère statistique et nous proposons notre algorithme incrémental.

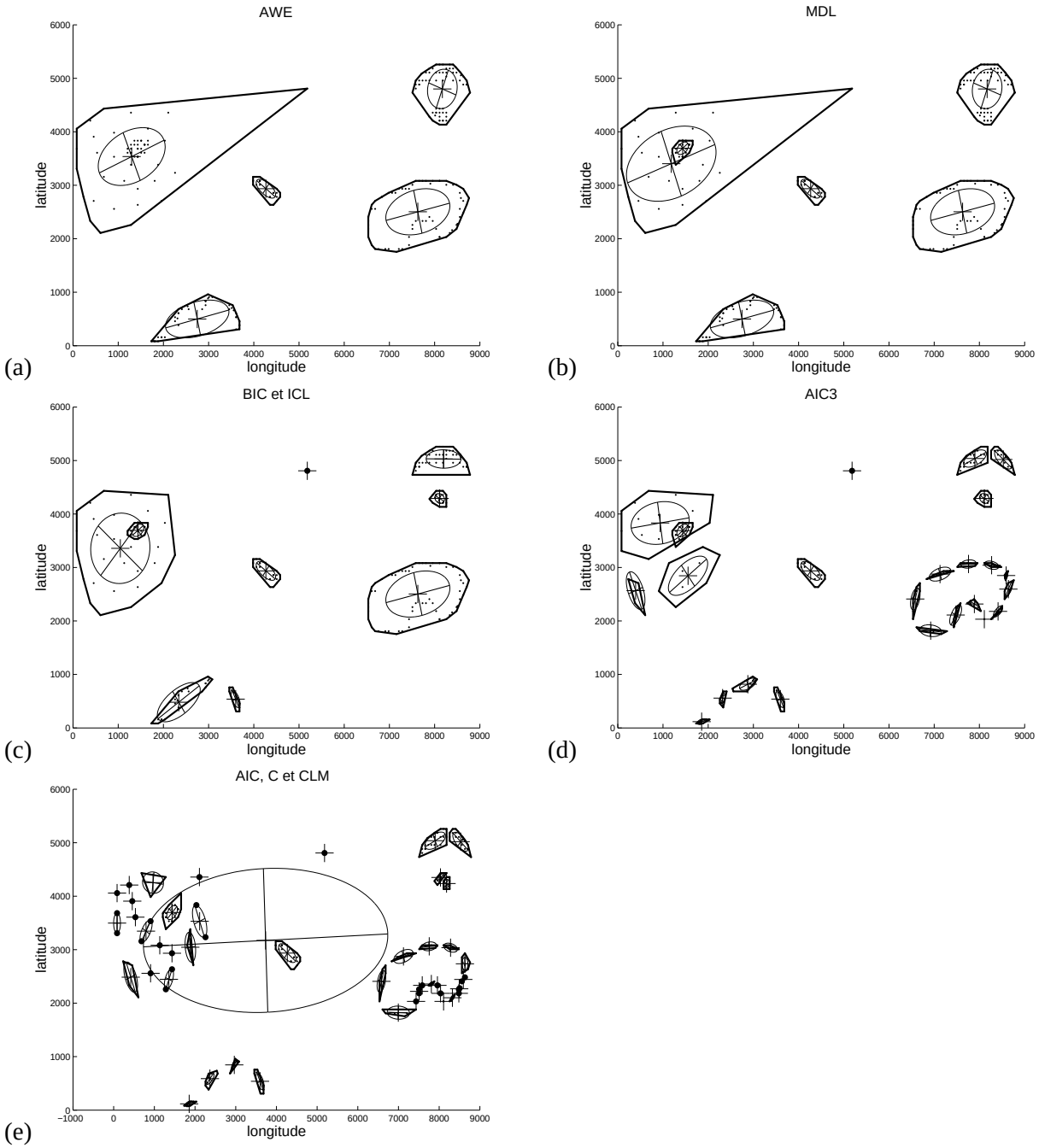


Figure 4.7: Les différentes partitions spatiales obtenues pour chaque critère statistique. Les composantes sont représentées par leur centre ('+') et leur variance (ellipse). Les polygones englobants représentent l'association des données aux composantes selon le critère MAP.

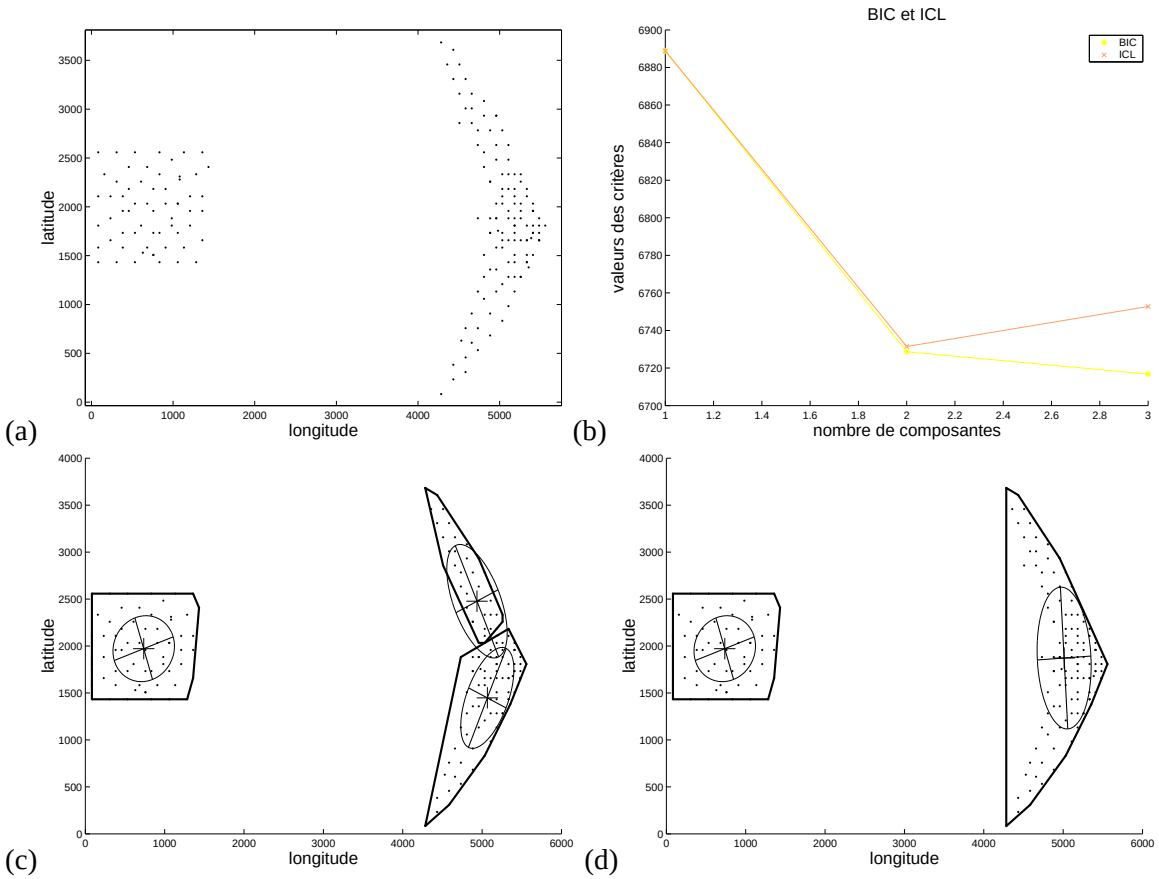


Figure 4.8: Comparaison entre le critère BIC et ICL : les données sont issues de tirages aléatoires selon des lois carré uniforme (à gauche) et courbe (à droite). Les critères BIC et ICL présentent respectivement un minimum pour 3 et 2 composantes (b). Les modèles obtenus sont représentés par les figures (c) et (d).

CHAPITRE 5

Algorithme incrémental d'optimisation du critère ICL

Nous détaillons dans ce chapitre plusieurs techniques pour obtenir un bon optimum de la vraisemblance. Nous proposons ensuite un algorithme incrémental et validons notre approche à l'aide d'expériences.

5.1 Introduction

Notre objectif est de fournir un système incrémental et automatique. L'algorithme doit ainsi permettre de faire évoluer la partition au fur et à mesure que de nouvelles données sont ajoutées dans la collection. Nous avons vu dans le chapitre précédent que le critère ICL est pertinent pour déterminer les paramètres et la complexité du modèle : il nous reste maintenant à proposer une technique de recherche dans l'espace de définition des paramètres pour obtenir une bonne optimisation de ce critère.

La recherche d'un optimum d'un critère statistique est un problème d'optimisation combinatoire : l'espace de définition des paramètres doit être exploré pour différentes complexités de modèles. Les paramètres étant définis sur un domaine continu, l'espace de recherche est grand. La recherche du meilleur optimum est ainsi impossible, et seul un optimum local peut être déterminé. Notre problème revient donc à trouver un compromis entre la complexité en temps de calcul et la qualité de l'optimisation.

Les algorithmes classiques détaillés dans le chapitre 3 (section (3.4)) présentent deux défauts majeurs : ils sont dépendants du choix des valeurs initiales des paramètres, ne permettant que d'obtenir un minimum local de la vraisemblance, et fonctionnent avec un nombre de composantes constant. Nous introduisons tout d'abord une technique classique pour résoudre ces deux problèmes, basée sur plusieurs stratégies proposées dans [14] permettant d'améliorer la recherche d'un optimum de la vraisemblance. Ensuite nous détaillons l'algorithme SMEM (*Split and Merge EM*) permettant de réaliser des sauts semi-locaux dans l'espace des paramètres afin de trouver une meilleure optimisation de la vraisemblance. Ces sauts sont obtenus à l'aide de fusions et divisions de composantes. Nous adaptons ensuite cet algorithme pour concevoir notre algorithme incrémental d'optimisation. Les propriétés et le comportement de notre algorithme sont explicités en détail. Enfin nous validons notre approche par des expériences sur des données artificielles réalistes et réelles.

5.2 Stratégies d'optimisation

Nous présentons dans cette section deux approches pour améliorer l'optimisation de la vraisemblance obtenue à l'aide d'algorithmes de type EM (présentés dans le chapitre 3).

5.2.1 Recherche de la meilleure initialisation

Plusieurs stratégies ont été étudiées dans [14] pour répondre au problème de la dépendance des algorithmes de type EM aux paramètres initiaux et ainsi améliorer l'optimisation de la vraisemblance. Elles sont fondées sur une recherche de la meilleure initialisation des valeurs des paramètres. Quatre stratégies ont été comparées :

1. initialisation aléatoire : *EM* ;
2. utilisation de l'algorithme CEM : *CEM-EM* ;
3. courtes itérations de l'algorithme EM : *em-EM* ;
4. utilisation de l'algorithme SEM : *SEMmean-EM* et *SEMmax-EM*.

La stratégie *EM* consiste à initialiser aléatoirement les paramètres du modèle et d'itérer l'algorithme EM jusqu'à convergence de la vraisemblance. Cette stratégie est la plus employée dans la littérature.

La stratégie *CEM-EM* consiste à itérer plusieurs fois l'algorithme CEM à partir de positions aléatoires, suivie d'itérations de l'algorithme EM à partir de la position ayant la meilleure optimisation de la vraisemblance complétée obtenue.

La stratégie *em-EM* est similaire à la précédente. On remplace tout simplement les itérations de l'algorithme CEM par de courtes itérations de l'algorithme EM. Une itération courte signifie que l'algorithme EM est arrêté avant la convergence de la vraisemblance. L'algorithme est stoppé quand la variation de la vraisemblance entre 2 itérations est inférieure à une limite définie manuellement. La stratégie consiste à appliquer de courtes itérations de l'algorithme EM à partir de positions aléatoires, suivies de longues itérations de l'algorithme EM à partir de la position avec la meilleure optimisation de la vraisemblance obtenue précédemment.

Enfin les stratégies *SEMmean-EM* et *SEMmax-EM* sont basées sur l'algorithme SEM. Elles sont définies par :

- *SEMmean-EM* : itérations de l'algorithme SEM suivies de l'algorithme EM initialisé avec la moyenne de la séquence des paramètres obtenus avec l'algorithme SEM. Il est supposé que l'algorithme SEM passe beaucoup de temps près des maxima de la vraisemblance ;
- *SEMmax-EM* : itérations de l'algorithme SEM suivies de l'algorithme EM à partir de la position avec la meilleure optimisation de la vraisemblance obtenue dans la séquence des paramètres produit par SEM. Il est supposé ici que la séquence de paramètres entre rapidement dans le voisinage du maximum global de la vraisemblance.

Une technique pour fournir un système automatique consiste donc à appliquer une de ces stratégies en faisant varier le nombre de composantes. Les modèles sont ensuite comparés avec un critère statistique, dans notre cas le critère ICL, et nous sélectionnons celui présentant la meilleure optimisation.

Les expériences proposées dans [14] ont montré que les différentes stratégies présentent des performances similaires. La stratégie *em-EM* se démarque néanmoins en étant légèrement meilleure et plus stable du point de vue de la similarité des partitions fournies (pour plusieurs essais sur un même jeu de données). Dans le chapitre précédent, les critères statistiques ont été comparés avec cette technique.

La technique consistant à utiliser une de ces stratégies en faisant varier la complexité du modèle présente néanmoins des inconvénients :

- elle n'est pas automatique puisque l'utilisateur doit fixer les limites inférieure et supérieure de la complexité des modèles ;
- elle n'est pas déterministe, plusieurs tests pouvant fournir des résultats différents (dus aux initialisations aléatoires). Le nombre d'itérations des algorithmes a tendance à limiter ce point, mais pour des données volumineuses ou mal structurées et un grand nombre de classes, il est difficile d'obtenir à chaque fois les mêmes résultats ;
- elle nécessite de recalculer entièrement le modèle pour chaque modification des données. Un système incrémental n'est donc pas réalisable avec cette approche. Cette méthode ne prend pas en compte l'avantage de disposer d'une partition initiale avant l'ajout de la nouvelle donnée ;
- la complexité en temps de calcul dépend du nombre d'itérations disponibles pour l'algorithme EM (court et long) et du nombre de modèles testés. Mais pour obtenir de bons résultats, il faut permettre un grand nombre d'itérations.

En conclusion, il nous faut proposer une autre approche pour concevoir notre algorithme incrémental.

5.2.2 L'algorithme SMEM

L'algorithme SMEM (*Split and Merge EM*) permet d'améliorer la vraisemblance d'un modèle optimisé, en réalisant des sauts semi-locaux dans l'espace des paramètres. Ces sauts sont obtenus en fusionnant ou divisant des composantes du modèle existant. L'algorithme d'optimisation à tendance à tomber dans un minimum local quand la disposition des composantes dans l'espace de définition des données n'est pas équilibrée (manque de composantes dans une partie comprenant beaucoup de données alors qu'une partie pauvre en données comprend plus de composantes). Les algorithmes décrits dans le chapitre précédent ne sont pas assez robustes face à ce problème : ils ne peuvent pas bouger de composantes d'une région surpeuplée vers une région sous-peuplée sans passer par des positions pénalisant la vraisemblance [115].

La figure 5.1 ci-après illustre ce point. Le jeu de données est composé de trois groupes suivant chacun une distribution gaussienne (un groupe de données en haut et deux en bas). La figure 5.1(a) présente le modèle initial et la figure 5.1(b) le "vrai" modèle. Nous avons calculé la vraisemblance du modèle en rapprochant au fur et à mesure une des deux composantes associée aux données du haut vers les données du bas. La variation de la vraisemblance est détaillée par la courbe de la figure 5.1(c). Les trois premiers minima locaux sont obtenus quand la composante est encore associée aux données du haut et le dernier est obtenu pour le modèle 5.1(b) : un algorithme de type EM ne peut donc passer d'une configuration

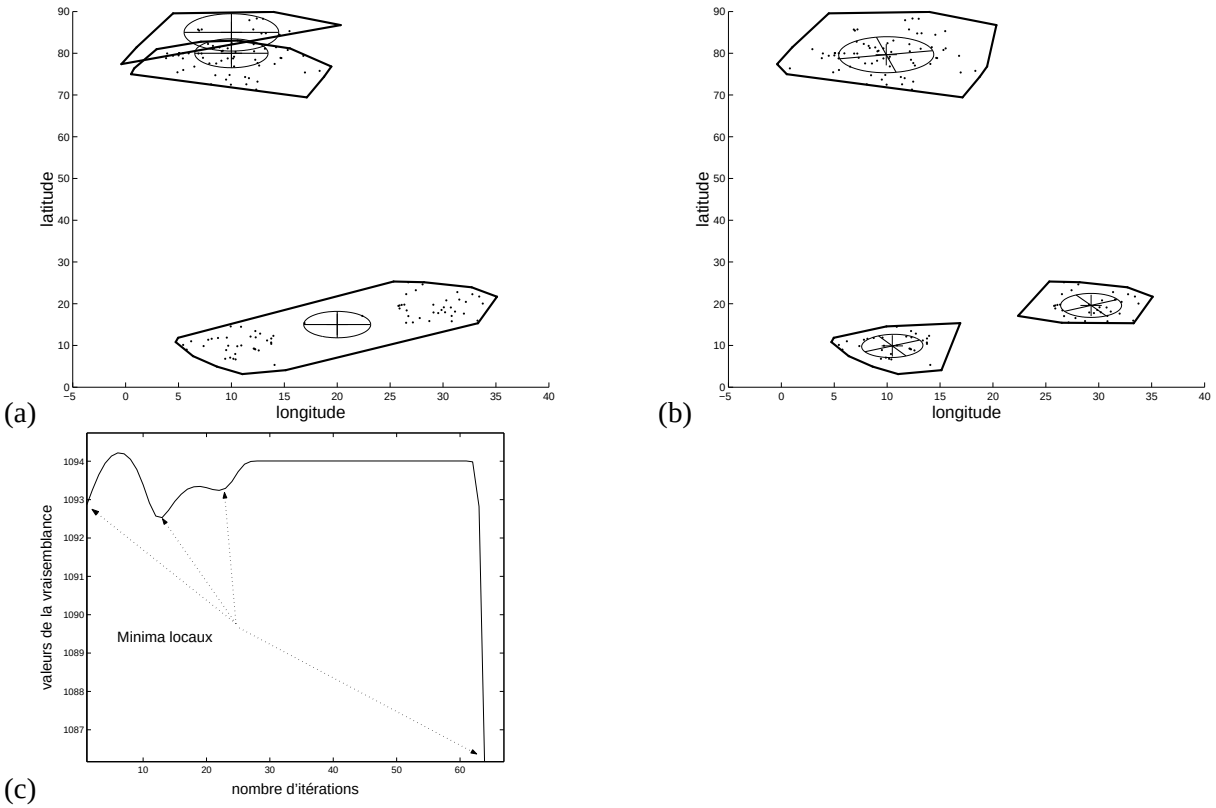


Figure 5.1: Illustration des minima locaux du critère de vraisemblance obtenus avec l'algorithme EM. Les figures (a) et (b) représentent respectivement le modèle initial et le "vrai" modèle. Les points sont les données et les composantes sont représentées par leur centre ('+') et leur variance (ellipse). Les polygones englobants représentent l'association des données aux composantes selon le critère MAP. La figure (c) représente la variation de la vraisemblance obtenue en "descendant" une composante associée aux données du haut vers celles du bas. Les trois premiers minima locaux sont associés à un modèle similaire au modèle (a) et le dernier est associé au modèle (b). Nous ne pouvons pas passer du modèle (b) au modèle (a) avec un algorithme de type EM puisqu'il ne peut pas sortir d'un minimum local.

à l'autre (du modèle 5.1(a) au modèle 5.1(b)) puisqu'il ne peut pas sortir d'un minimum local de la vraisemblance.

La division ou fusion de composantes tentent de régler le problème en permettant des sauts dans l'espace des paramètres. Il faut bien noter que le nombre de composantes est constant : à chaque itération, une composante est divisée et deux autres sont fusionnées.

Nous présentons tout d'abord les critères de fusion et division, proposés par Ueda et al. [115], l'initialisation des paramètres et un algorithme EM partiel pour mettre à jour le modèle localement. Enfin nous détaillons l'algorithme SMEM.

Critère de fusion et de division

Le nombre de possibilités de fusions et divisions étant élevé, des critères sont utilisés pour ordonner les candidats. Le critère de fusion proposé dans [115] est basé sur les probabilités *a posteriori* des composantes. L'idée est de comparer deux à deux ces probabilités des composantes et de fusionner celles

présentant le plus de similarité :

$$J_{fusion}(i,j,\theta) = \frac{\langle t_i^t, t_j \rangle}{||t_i|| \cdot ||t_j||}, \tag{5.1}$$

où t_i représente les probabilités d'affectation *a posteriori* des données à la composante i (la colonne i de la matrice t). Cette technique permet de fusionner les composantes avec des paramètres proches. Les composantes avec un score J_{fusion} élevé sont fusionnées en priorité.

Le critère de division est basé sur une divergence locale de Kullback-Leibler entre la densité des données locales $f_k(x,\theta)$ autour de la composante k et la densité de la composante k spécifiée par les paramètres estimés θ. L'idée est ici de diviser les composantes dans les espaces où le modèle contient peu de composantes. Cette divergence est définie par :

$$J_{division}(k,\theta) = \int f_k(x,\theta) log \frac{f_k(x,\theta)}{p_k(x,\theta_k)} dx, \tag{5.2}$$

où la densité des données locales $f_k(x,\theta)$ est définie par :

$$f_k(x,\theta) = \frac{\sum_{i=1}^n \delta(x - x_i) P(k|x_n,\theta)}{\sum_{i=1}^n P(k|x_i,\theta)}, \tag{5.3}$$

où δ est la fonction delta de Dirac.

Une composante avec un score $J_{division}$ élevé est susceptible d'être divisée. Les candidats sont ensuite triés, d'abord en fonction des candidats à la fusion puis en fonction des candidats à la division.

Initialisation des paramètres

L'initialisation des paramètres θ pour les modèles obtenus après une fusion ou une division de composantes est définie comme suit. Les paramètres initiaux après une fusion de deux composantes i et j en une composante i' sont :

$$p_{i'} = p_i + p_j \quad \text{et} \quad \theta_{i'} = \frac{p_i \cdot \theta_i + p_j \cdot \theta_j}{p_i + p_j} \tag{5.4}$$

Pour la division d'une composante k en deux composantes j' et k' :

$$p_{j'} = p_{k'} = \frac{p_k}{2}, \quad \mu_{j'} = \mu_k + \epsilon, \quad \mu_{k'} = \mu_k + \epsilon' \quad \text{et} \tag{5.5}$$

$$\Sigma_{j'} = \Sigma_{k'} = det(\Sigma_k)^{(1/d)} / I_d, \tag{5.6}$$

où ϵ est un vecteur de faible norme aléatoire, $det(\Sigma)$ le déterminant de Σ et I_d la matrice d'identité.

EM partiel

Après la phase d'initialisation, l'algorithme SMEM utilise un algorithme partiel pour optimiser localement les paramètres des composantes modifiées sans affecter les autres. Les probabilités *a posteriori* du nouveau modèle sont donc localement mises à jour pendant l'étape d'estimation de l'algorithme EM. L'étape E est ainsi remplacée par :

$$P(m|x,\Theta) = \frac{p_m \cdot \mathcal{N}(x,\theta_m)}{\sum_{l=i,j,k} p_l \cdot \mathcal{N}(x,\theta_l)} \cdot \sum_{m=i,j,k} P(m|x,\theta), \tag{5.7}$$

Algorithme 1 algorithme d'optimisation SMEM

1.optimisation du modèle initial $\mathcal{M}_1$. Nous notons L_1 la log-vraisemblance initiale ;
2.calcul des critères de fusion (equation 5.1) et de division (équation 5.2), puis classement des 2-uplets $((i,j),k)$, où i et j sont fusionnés et k divisé ;
3.phase de division et fusion :
pour les α premiers 2-uplet (pratiquement $\alpha = 5$) **faire**
  3.1.fusionner les composantes i et j, diviser la composante k et mettre à jour le modèle à partir de $\mathcal{M}_1$;
  3.2.itérer l'algorithme partiel EM jusqu'à convergence ;
  3.3.itérer l'algorithme EM classique jusqu'à convergence. Nous obtenons un modèle $\mathcal{M}_2$ avec un critère de vraisemblance L_2 ;
  si $L_2 < L_1$ **alors**
    3.4.$\mathcal{M}_1 \leftarrow \mathcal{M}_2$;
    3.5.$L_1 \leftarrow L_2$ et aller à l'instruction 2.
  finsi
fin pour

pour $m = i, j, k$. On itère ensuite l'algorithme EM classique jusqu'à convergence.

Algorithme SMEM

Une itération de l'algorithme SMEM consiste à tenter d'améliorer la vraisemblance optimisée d'un modèle en testant une fusion et une division de composantes. Pour chaque test, nous retenons le nouveau modèle si la vraisemblance est améliorée. L'algorithme 1 ci-dessus présente en détail l'algorithme. Chaque fois qu'un nouveau modèle est retenu, les critères de fusions et divisions sont re-calculés et les α premiers candidats de cette liste sont testés. Le paramètre α définit ainsi le nombre maximum d'itérations de fusions et de divisions. Il est initialisé à 5, un nombre supérieur n'étant pas nécessaire d'après les résultats expérimentaux présentés dans [115]. L'algorithme s'arrête quand les α premiers candidats n'améliorent plus la vraisemblance.

Une limite de cet algorithme est que le nombre de composantes dans le modèle reste constant. Néanmoins l'optimisation d'un modèle en proposant des fusions et divisions de composantes est une idée intéressante pour concevoir un système automatique. Nous nous basons sur une approche similaire pour proposer un algorithme incrémental, où le nombre de composantes varie en fonction de la structure des données.

5.3 Proposition d'un algorithme incrémental

Nous avons trouvé un seul algorithme incrémental basé sur les modèles de mélange dans la littérature. Ces travaux, proposés par Zivkovic et Heijden [127], sont similaires aux algorithmes stochastiques EM mais permettent d'ajouter les données incrémentalement. Les paramètres du modèle sont initialisés avec une complexité supérieure au nombre réel de composantes. Au fur et à mesure des ajouts de nouvelles données, les paramètres sont mis à jour et les composantes associées à de petits échantillons sont supprimées. Les inconvénients de cette approche sont le réglage manuel du nombre initial de composantes et la technique de sélection de la complexité du modèle (dans notre contexte, une composante doit pouvoir être associée à de petits échantillons). Notons que Neal et Hilton ont aussi proposé un al-

gorithme EM dit "incrémental" [85]. Il consiste à appliquer l'algorithme EM seulement sur une partie des données à chaque itération. Aucun de ces deux algorithmes ne convient pour notre application. Aussi nous proposons notre propre algorithme incrémental.

5.3.1 Description de notre algorithme incrémental d'optimisation

La recherche de partitions de données en temps réel (après chaque ajout successif d'une nouvelle donnée) suppose de pouvoir mettre à jour à faible coût l'affectation des données aux classes et d'ajuster le nombre de composantes en fonction des nouvelles données. Notre proposition consiste à utiliser la partition obtenue au temps t comme initialisation de l'optimisation du critère ICL pour la partition à $t + 1$: cela fournit immédiatement une correspondance explicite entre les classes entre deux instants successifs. La stabilité de la partition est favorisée au cours du temps et cela facilite son exploration par l'utilisateur dans une IHM graphique. Deux problèmes sont néanmoins à résoudre :

1. le nombre de composantes doit évoluer automatiquement : au fur et à mesure que les données sont ajoutées, les paramètres des composantes et leur nombre doivent être mis à jour ;

2. le flux de données ne peut pas être modélisé comme une série de données indépendantes générée à partir d'une loi de probabilité fixée. En effet, un utilisateur prend généralement ces images par paquets [81] : les images appartenant au même paquet sont susceptibles d'appartenir aux mêmes événements. Comme nous l'expliquons dans le chapitre 2 (section 2.4.1, page 30), les données présentent une dépendance entres elles. Au fur et à mesure que les données d'un paquet sont ajoutées, les paramètres du modèle sont optimisés localement sur ces données et il peut être plus difficile de les mettre à jour si un nouvel événement apparaît. L'optimisation au cours du temps est ainsi délicate et des minima locaux sont souvent obtenus si seul un algorithme EM classique est utilisé.

Notre solution permet de résoudre, au moins partiellement, ces deux points. Nous proposons une procédure d'optimisation basée sur des sauts semi-locaux dans l'espace des paramètres en appliquant des fusions et divisions de composantes. Nous alternons les phases de sauts avec des itérations de l'algorithme EM, jusqu'à convergence. Les deux étapes permettent de minimiser le critère ICL et servent le même objectif : éviter les minima locaux et permettre l'évolution du nombre de composantes au cours du temps. Les différences avec les travaux similaires de Ueda [115] sont significatives, ceux-ci ne présentent pas d'aspect incrémental et gardent le nombre de composantes constant.

Critères de fusion et de division

Les critères de fusion et division proposés dans [115] ne sont pas adaptés à notre cas d'utilisation :

- le critère de fusion précédent est basé sur les probabilités *a posteriori* d'affectation des données aux composantes. L'idée est de comparer les composantes deux à deux à l'aide de ces probabilités (les colonnes de la matrice t définie dans la section 3.4, page 42) et de fusionner celles présentant le plus de similarité. On fusionne ainsi les composantes avec les paramètres les plus proches. Cette technique n'est pas pertinente dans le cas de petits échantillons. Par exemple, si on compare deux composantes proches associées chacune à une seule donnée, elle ne seront pas considérées

comme de bons candidats (puisque le score J_{fusion} est nul). Dans notre contexte, cette situation est fréquemment rencontrée (au début du processus de classification par exemple, quand on dispose de peu de données) ;

- pour le critère de division, l'idée de diviser les composantes dans les espaces où le modèle contient peu de composantes ne nous semble pas pertinente. Nous préférons nous baser sur un critère de classifiabilité pour déterminer les composantes définissant mal leur données associées.

Nous proposons donc deux nouveaux critères. Nous proposons de se baser sur la distance de Mahalanobis pour ordonner les candidats à la fusion :

$$J_{merge}(i,j,\theta) = min\{D(\mu_i,\Sigma_i,\mu_j), \\ D(\mu_j,\Sigma_j,\mu_i)\}, \tag{5.8}$$

où $D(\mu_j,\Sigma_j,\mu_i) = (\mu_i-\mu_j\}^T \cdot \Sigma_j^{-1} \cdot (\mu_i-\mu_j)$. Dans le cas de petits échantillons, cette distance ne pose pas de problème pour être calculée puisque notre régularisation des matrices de variance garantit un volume minimum aux composantes. Les composantes avec un score J_{merge} faible sont fusionnées.

Le critère de division est basé sur un critère d'entropie obtenu pour chaque composante individuellement. Nous préférons nous baser sur un critère de classifiabilité plus pertinent pour garantir la qualité de la classification. Nous rappelons qu'une composante avec une entropie forte suggère que ses paramètres se chevauchent avec une ou plusieurs autres composantes proches : il existe un ensemble de données qui serait "commun" à ces composantes (faiblement affectées à chacune d'entres elles). Nous proposons d'essayer de rajouter une nouvelle composante pour représenter cet ensemble commun. Dans le cas où cet ensemble est légèrement isolé des autres données associées à ces composantes, une nouvelle composante est pertinente. Notre critère de division est donc défini par :

$$J_{split}(k,\Theta) = -\sum_{i=1}^{n} t_{ik} \cdot log(t_{ik}) \tag{5.9}$$

Une composante avec un score J_{split} élevé est susceptible d'être divisée.

Initialisation des paramètres

L'initialisation des paramètres θ pour les modèles obtenus après une fusion ou une division de composantes est pratiquement identique à l'algorithme SMEM. Seule l'initialisation des centres après une division présente une petite différence sur le paramètre ϵ. Nous définissons ce paramètre comme un vecteur de faible norme colinéaire à la plus grande variance de la composante divisée. Une division entre deux composantes est ainsi réalisée en gardant l'orientation de la composante initiale. La figure 5.2 ci-après présente un exemple d'initialisation des paramètres après une division ((a)→(b)) et une fusion ((c)→(d)).

EM partiel

Après l'initialisation nous procédons également à une mise à jour locale des nouvelles composantes à l'aide de l'algorithme EM partiel. Par contre, il n'est appliqué que pour le cas où une composante est divisée. En effet, pour une fusion, seule une composante doit être mise à jour : l'algorithme EM partiel est inefficace, le terme de gauche de l'équation (5.7) étant égal à 1.

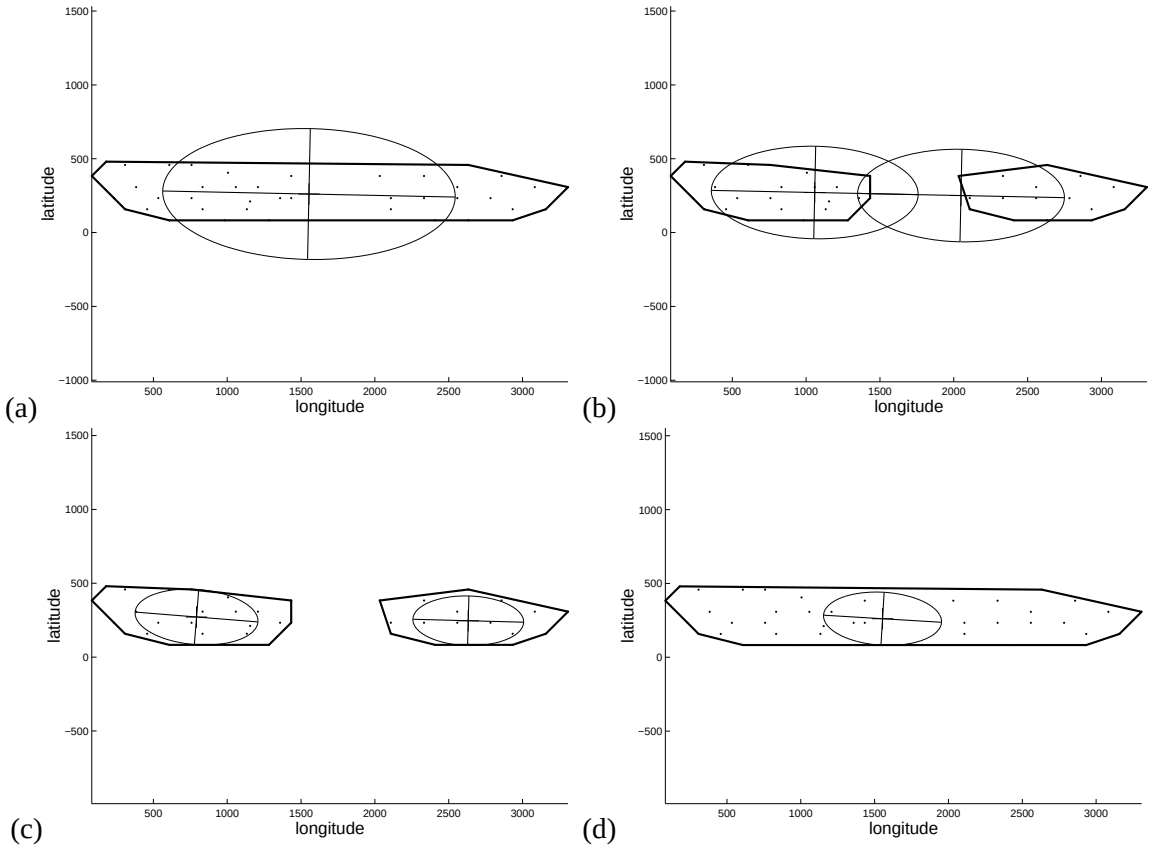


Figure 5.2: Initialisation des paramètres après une fusion ou une division de composantes : la figure (b) représente les nouveaux paramètres du modèle après une division du modèle (a). La figure (d) est le modèle obtenu après la fusion des deux composantes du modèle (c).

Après une division, les paramètres des nouvelles composantes sont d'abord mis à jour localement avant d'itérer l'algorithme EM sur l'ensemble des composantes. Pour une composante divisée en deux composantes i et j, les probabilités *a posteriori* d'affectation des données aux composantes sont calculées avec l'équation (5.7) pour $m = i, j$.

Algorithme 2 algorithme incrémental d'optimisation du critère ICL

1.ajouter une nouvelle donnée dans l'ensemble X et itérer l'algorithme EM sur le modèle initial.
Soit ICL_1 le critère ICL obtenu à la convergence et $\mathcal{M}_1$ le modèle correspondant ;
2.suppression des composantes vides ;
3.phase de division : ordonner les candidats à diviser selon le critère (5.9) ;
pour les α premières composantes ordonnées par entropie décroissante (pratiquement $\alpha = 5$) **faire**
  3.1.diviser la composante et mettre à jour le modèle à partir de $\mathcal{M}_1$;
  3.2.itérer l'algorithme EM partiel ;
  3.3.itérer l'algorithme EM jusqu'à convergence. On obtient un modèle $\mathcal{M}_2$ avec un critère ICL_2 ;
  si $ICL_2 < ICL_1$ **alors**
    3.4.$\mathcal{M}_1 \leftarrow \mathcal{M}_2$;
    3.5.$ICL_1 \leftarrow ICL_2$;
    3.6.aller à l'étape 2 ;
  finsi
fin pour
4.suppression des composantes vides ;
5.phase de fusion : ordonner les paires de composantes à la fusion en fonction du critère (5.8) ;
pour les α premières composantes ordonnées selon une distance croissante de Mahalanobis **faire**
  5.1.fusionner les composantes et mettre à jour le modèle à partir de $\mathcal{M}_1$;
  5.2.itérer l'algorithme EM partiel ;
  5.3.itérer l'algorithme EM jusqu'à convergence. On obtient un modèle $\mathcal{M}_2$ avec un critère ICL_2 ;
  si $ICL_2 < ICL_1$ **alors**
    5.4.$\mathcal{M}_1 \leftarrow \mathcal{M}_2$;
    5.5.$ICL_1 \leftarrow ICL_2$;
    5.6.aller à l'étape 3 ;
  finsi
fin pour
6.suppression des composantes vides.

Algorithme EM

Notre algorithme EM d'optimisation incrémental consiste à tester plusieurs divisions de composantes suivies de plusieurs fusions pour chaque nouvelle donnée ajoutée. Nous retenons, pour chaque modification du modèle, le modèle optimisant le critère ICL (4.18). L'algorithme 2 ci-après détaille notre approche. Le paramètre α définit le nombre maximum d'itérations de fusions et de divisions. Comme pour l'algorithme SMEM, ce paramètre est initialisé à 5. Quand une fusion ou une division est validée (quand le critère ICL est amélioré), nous re-calculons la liste des candidats et testons les α premiers candidats de cette liste. Nous présentons maintenant plusieurs points permettant de bien comprendre notre algorithme.

5.3.2 Propriétés de notre algorithme

Nous présentons dans cette section plusieurs propriétés sur le comportement de notre algorithme et nos choix de conception.

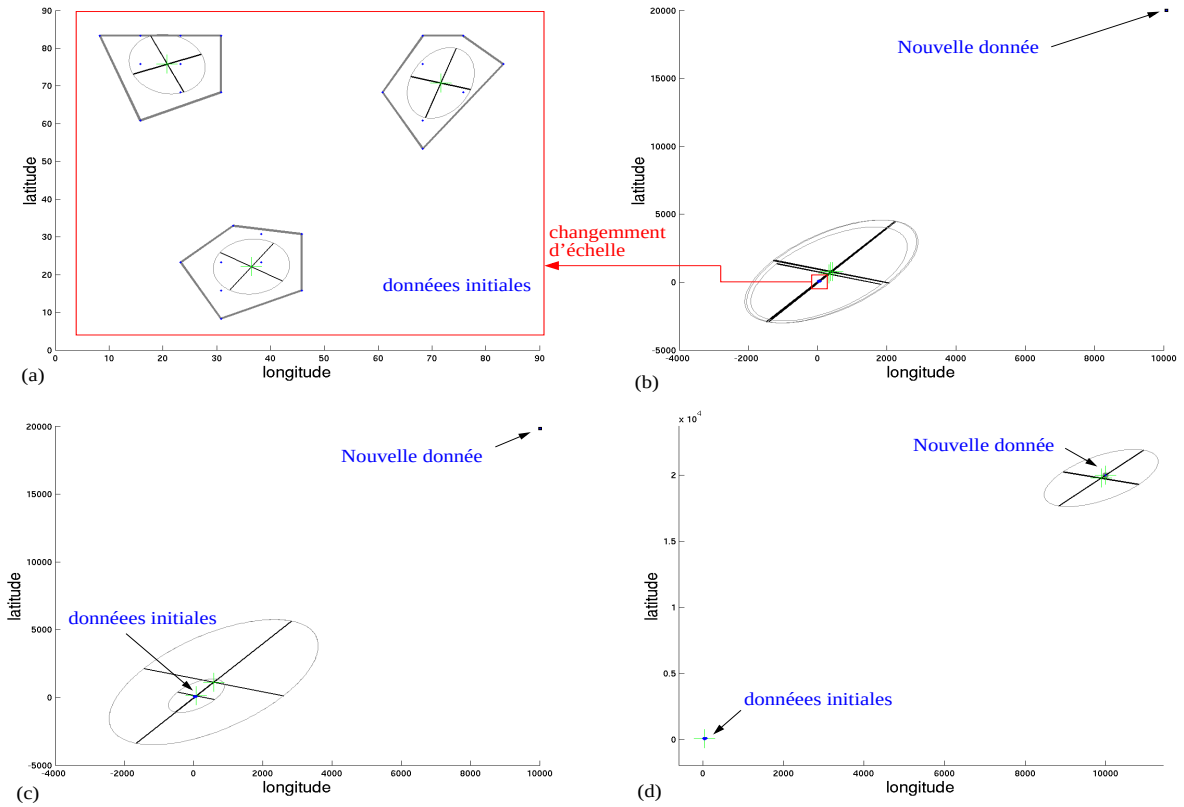


Figure 5.3: cet exemple présente le comportement de notre algorithme dans le cas où les composantes sont associées à peu de données et qu'une nouvelle donnée entraîne un changement d'échelle. La figure (a) est le modèle initial. Une nouvelle donnée est ajoutée en $[10000, 20000]$ *et l'algorithme EM est itéré une seule fois. Le modèle (b) présente le résultat obtenu (l'espace de définition des données initiales est signalé par un carré rouge). Les figures (c) et (d) présentent l'évolution de la partition au fur et à mesure des itérations de l'algorithme EM jusqu'à convergence.*

Impact sur les paramètres lors d'un changement d'échelle

L'ajout d'une nouvelle donnée peut quelquefois entraîner beaucoup de modifications dans le modèle. Notamment dans le cas où les composantes sont associées à peu de données et qu'un changement d'échelle à lieu. Nous parlons de *changement d'échelle* quand la nouvelle donnée est très éloignée des données précédentes.

L'étape E de l'algorithme EM consiste à calculer l'espérances des données aux composantes. La nouvelle donnée étant éloignée de ces dernières, elle est affectée à l'ensemble des composantes avec une espérance à peu près similaire pour chacune d'entres elles. L'étape M consiste ensuite à calculer les nouveaux paramètres à partir des espérances précédemment obtenues. Si chaque composante est associée à peu de données, leurs paramètres varient fortement en raison de l'espérance de la nouvelle donnée.

La figure 5.3 ci-dessus présente un exemple d'ajout d'une nouvelle donnée provoquant un changement d'échelle. La figure 5.3(a) présente le modèle initial. Après l'ajout de la nouvelle donnée et une itération de l'algorithme EM, les paramètres du modèle sont fortement modifiés, comme le montre la

figure 5.3(b). La nouvelle donnée est ajoutée en $[10000; 20000]$ et son association à toutes les composantes durant l'étape E entraîne un déplacement des centres et une augmentation des variances de chaque composante. Dans notre exemple, ces paramètres se retrouvent décalés en direction de la nouvelle donnée. La figure 5.3(c) présente le modèle après plusieurs itérations de l'algorithme EM et la figure (d), le modèle obtenu après convergence. Deux des composantes se déplacent lentement vers la nouvelle donnée, la composante la plus large sur (c) est en fait une superposition de deux composantes. Finalement, la figure (d) présente le modèle obtenu après convergence. Les données initiales et la nouvelle donnée sont associées chacune à une composante. La troisième composante présente une large variance englobant la composante de la nouvelle donnée. En pratique, c'est une classe vide qui serait effacée avec notre algorithme d'optimisation.

Un tel cas de figure est néanmoins pertinent puisque le changement d'échelle est correctement géré. Dans notre exemple, le modèle final, obtenu après avoir supprimé la composante vide, représente bien la structure des données. Nous rappelons qu'une nouvelle donnée n'entraîne autant de modifications que si les composantes sont associées à peu de données. Dans le cas contraire, la nouvelle donnée n'a que peu d'impact sur les composantes et le modèle restera stable. Le changement d'échelle sera alors géré avec des fusions de composantes si le critère ICL le permet.

Stratégies d'optimisation du critère ICL

Plusieurs solutions étaient possibles pour la première étape de notre algorithme, après l'ajout d'une nouvelle donnée dans le modèle :

- comparer le critère ICL du modèle initial (avant toute optimisation et en prenant en compte la nouvelle donnée) et le modèle optimisé (après optimisation avec l'algorithme EM en prenant en compte la nouvelle donnée). Le modèle minimisant le critère ICL est choisi et des fusions et divisions de composantes sont ensuite réalisées ;

- optimiser le modèle et ensuite tester des fusions et des divisions de composantes sur le modèle obtenu. C'est la solution que nous avons choisie.

La première solution consiste à toujours choisir le modèle qui optimise le critère ICL : après avoir ajouté la nouvelle donnée, on compare les critères ICL du modèle initial non optimisé et du modèle optimisé ; on choisit le modèle optimisant le critère ICL et on teste ensuite des fusions et divisions de composantes sur ce modèle. Mais un problème se pose : dans le cas où le modèle retenu est le modèle initial (le modèle non optimisé après l'ajout de la nouvelle donnée), le choix des composantes à fusionner ou diviser à l'aide de nos critères n'est plus efficace. En effet, les critères de sélection seraient calculés sur un modèle ne prenant pas en compte la nouvelle donnée : les composantes susceptibles d'être divisées ou fusionnées ne seront ainsi pas obligatoirement les plus pertinentes.

Nous avons donc choisi de prendre en compte la nouvelle donnée dès les premiers pas de l'algorithme, en optimisant directement le modèle initial. Une fois le modèle mis à jour, et les composantes vides supprimées, les critères de fusion et de division sont calculés à partir d'un modèle prenant en compte l'ajout de la nouvelle donnée, ce qui a plus de sens pour pouvoir trouver leur nouvelle structure.

Il faut noter que le choix de la première solution ne permet pas de régler le problème d'optimisation quand les composantes sont associées à peu de données et qu'un changement d'échelle a lieu (voir l'exemple sur la figure 5.3 page 89). Même dans le cas où le modèle initial est sélectionné (en ne prenant pas en compte la nouvelle donnée), les itérations de l'algorithme EM sur les nouveaux paramètres ob-

tenus pour tester la première division d'une composante peuvent modifier entièrement le modèle, si ses composantes sont associées à peu de données.

Gestion des composantes vides

Étant dans un contexte de classification, chaque composante n'est utile que si elle est associée à un ensemble non vide de données, selon le critère MAP. Après chaque modification du modèle, les itérations de l'algorithme EM ou un test d'une fusion ou d'une division de composantes, nous recherchons et supprimons les composantes vides (étapes 2, 4 et 6 de l'algorithme 2).

Dépendance à l'ordre des données

Notre algorithme est bien entendu dépendant de l'ordre des données. L'ordre peut avoir un fort impact suivant les changements d'échelle. Si l'espace de définition des données est directement très grand avec les premières données, nous avons plus de chance d'obtenir un modèle présentant peu de détails, composé de larges classes. Dans le cas où l'espace de définition s'accroît petit à petit avec l'ajout des nouvelles données, nous pouvons obtenir un modèle détaillé (si un changement d'échelle ne provoque pas ensuite une perte de détail).

Selon l'ordre des données, nous n'obtenons pas le même optimum local du critère ICL.

Notre approche fournit un procédé incrémental basé sur l'optimisation du critère ICL. Nous présentons dans la section suivante des expériences pour valider notre approche.

5.4 Classification de trois collections d'images à l'aide de notre algorithme incrémental d'optimisation

Nous validons notre approche en appliquant notre algorithme incrémental d'optimisation sur trois jeux de données différents.

5.4.1 Collection artificielle

L'objectif de cette expérience est de comparer les résultats obtenus avec notre algorithme d'optimisation et ceux obtenus avec la technique *em-EM*. Les données spatiales et temporelles sont les mêmes que celles utilisées pour les expériences du chapitre précédent.

Classification temporelle

La figure 5.4(a) ci-après présente le résultat obtenu pour la classification temporelle. Le modèle est composé de 15 classes et la plupart d'entres elles présentent des similarités avec le modèle obtenu au chapitre précédent (figure 4.5(b), page 72). Le nombre de composantes reste du même ordre puisque nous obtenions 11 composantes et nous avons noté que 4 composantes sont similaires. Le modèle obtenu avec notre algorithme d'optimisation est légèrement plus détaillé. Les composantes sont bien définies pour la plupart, présentant une solution alternative au modèle précédent. Le seul défaut est lié à une sur-segmentation sur deux intervalles : $[2 \cdot 10^5; 4 \cdot 10^5]$, $[8 \cdot 10^5; 10 \cdot 10^5]$.

Nous obtenons une meilleure optimisation du critère ICL ($6, 45 \cdot 10^3$) avec notre algorithme que dans les expériences du chapitre précédent ($6, 53 \cdot 10^3$). Ce point montre que notre méthode est pertinente pour

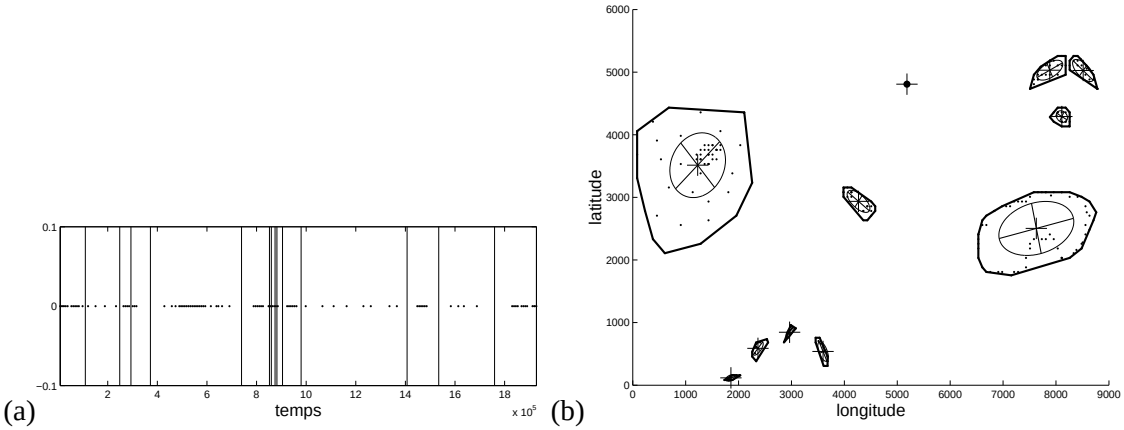


Figure 5.4: Collection artificielle : classification temporelle (a) et spatiale (b). Les points représentent les métadonnées spatiales ou temporelles. Les lignes (a) ou les polygones englobants (b) représentent les limites des classes selon le critère MAP. Les "+" et les ellipses indiquent respectivement les centres et les variances des composantes.

trouver un bon minimum local du critère ICL. Notons que ce résultat était attendu puisque la recherche dans l'espace des paramètres est beaucoup plus poussée avec notre approche.

Classification spatiale

La classification spatiale obtenue est présentée par la figure 5.4(b) ci-après. Le modèle contient 10 composantes, une de plus que le modèle obtenu avec la technique *em-EM* (figure 4.7(c), page 76). Les deux modèles sont similaires, 5 composantes étant identiques. La partition représente bien la structure des données : les composantes sont distinctes et mettent bien en valeur les différents lieux.

L'optimisation du critère ICL est meilleure que celle obtenue avec la technique *em-EM* : le critère ICL est de $8,54 \cdot 10^3$ avec notre algorithme et de $8,56 \cdot 10^3$ avec la technique précédente. Comme précédemment, notre algorithme est plus performant pour optimiser le critère ICL.

5.4.2 Collection artificielle réaliste

N'ayant pas à notre disposition un grand nombre de collections d'images, nous avons créé manuellement une collection réaliste, à partir d'une carte géographique pour les données spatiales et d'un calendrier pour les données temporelles, afin qu'elle semble vraisemblable. Nous appelons cette collection *la collection artificielle réaliste*. Pour cette expérience, nous disposons de 451 métadonnées d'images. Les figures 5.5(a) et (b) ci-contre présentent respectivement les données temporelles et spatiales. Les données temporelles présentent 5 groupes de données, dont le deuxième est fortement mal structuré. Les données spatiales sont composées de 4 groupes, chacun divisé en plusieurs sous-groupes.

Classification temporelle

La classification temporelle obtenue, présentée par la figure 5.7(a) page 97, est composée de 19 classes. La figure 5.7(b) fournit un zoom sur la partie gauche des données.

Les groupes distincts sur la partie droite ont été convenablement trouvés. La partie gauche est quant

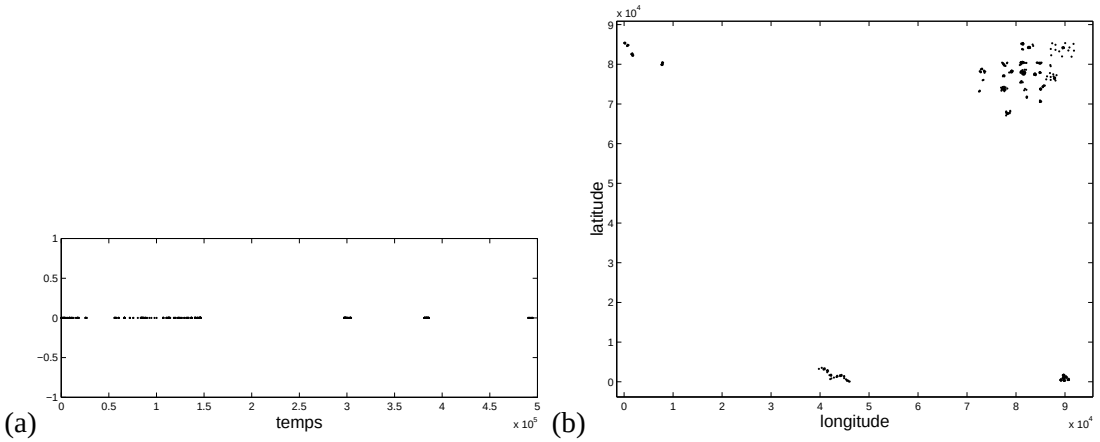


Figure 5.5: Collection artificielle réaliste : les figures (a) et (b) représentent respectivement les métadonnées temporelles et spatiales.

à elle plus difficile à évaluer, les données y étant moins structurées. La majorité des groupes semble néanmoins correcte. Les limites entres les classes sont visuellement justifiées (notamment dans la partie droite de la figure 5.7(b)), mais certaines composantes ne semblent pas assez détaillées. Par exemple sur la figure 5.7(b), la première composante est bien délimitée mais présente moins de détails par rapport aux autres : la structure de ses données associées n'est pas mise en valeur.

Afin de tester si de telles composantes étaient dues à une mauvaise optimisation du critère ICL, nous avons classé les données associées à ces groupes localement avec notre algorithme d'optimisation et inclus ensuite le modèle obtenu dans le modèle général. Le critère ICL était supérieur à celui obtenu initialement. Le problème ne semble donc pas venir de la procédure d'optimisation, mais plutôt du critère ICL qui ne permet pas l'ajout de nouvelles composantes.

Stabilité des classifications

Afin d'étudier l'évolution des partitions au fur et à mesure de l'ajout des nouvelles données, nous présentons dans la figure 5.8 page 98 les classifications obtenues pour 90 (a), 180 (b), 270 (c) et 360 (d) données. Les classifications obtenues semblent cohérentes, la structure des données étant correctement mise en valeur dans chaque figure. Nous notons que la première classe temporelle obtenue dans le modèle final était tout d'abord divisée en plusieurs composantes. L'intervalle $[0; 0,4 \cdot 10^5]$ sur les 4 figures est toujours divisé en plus d'une classe alors que dans le modèle final, nous n'en obtenons qu'une seule. Le changement d'échelle, dû au trois dernières classes compactes, a entraîné une perte de détail dans les premières classes temporelles : les composantes ont été fusionnées.

Classification spatiale

La classification spatiale obtenue, composée de 29 classes, est présentée par la figure 5.9(a) page 99. Des zooms de ce modèle sont proposés dans les figures 5.9(b), (c) et (d). La partition semble correcte : sur chacune des figures nous pouvons noter que les composantes sont bien délimitées. Nous obtenons généralement des composantes compactes mais aussi quelques unes associées à des données isolées éparpillées. La composante en haut à droite sur la figure 5.9(b) est associée à peu de données très étalées.

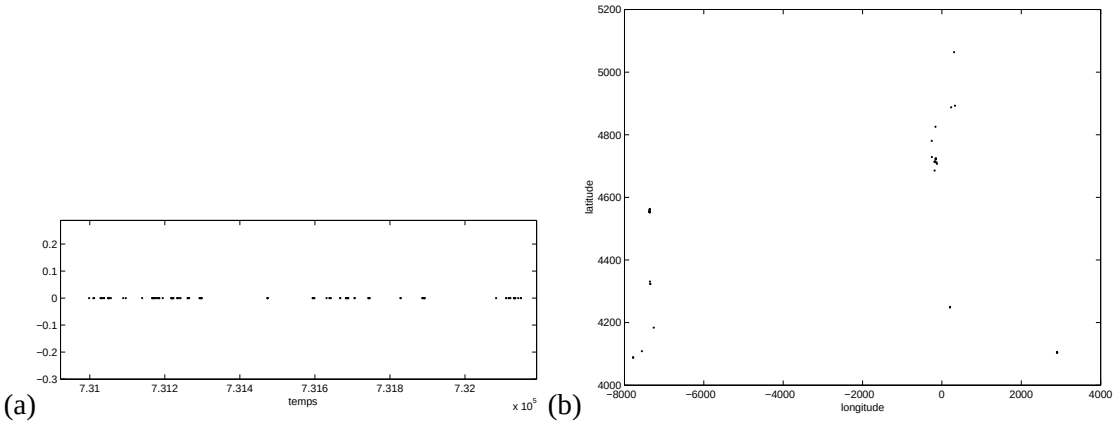


Figure 5.6: Collection réelle : les figures (a) et (b) représentent respectivement les métadonnées temporelles et spatiales.

Stabilité des classifications

Plusieurs partitions obtenues au cours du processus de classification sont présentées par la figure 5.10, page 100. Les résultats intermédiaires montrent que la partition évolue bien en fonction des ajouts des nouvelles données. Les premières données ajoutées appartiennent au groupe de données représenté par la figure 5.9(b). Malgré le changement d'échelle visible dans la figure 5.10(g), les premières composantes obtenues ont été en partie conservées. Elles étaient associées à un nombre assez élevé de données pour ne pas être modifiées.

Détail de la phase d'optimisation

Nous présentons maintenant un exemple sur la phase d'optimisation du critère ICL après l'ajout d'une nouvelle donnée. Les étapes successives de mise à jour de la partition sont présentées par les figures 5.11 et 5.12, page 101. La figure 5.11(haut) montre la partition initiale après l'ajout de la nouvelle donnée. Cette partition est peu pertinente, une des données isolées en bas étant associée à la plus large des composantes, alors que celle-ci n'est pas la plus proche. La figure 5.11(bas) présente la variation du critère ICL au cours des itérations de l'algorithme EM et des fusions et divisions de composantes. Après chaque phase de fusions et divisions, débutant aux itérations 100, 200, 300 et 400, l'algorithme EM est itéré 100 fois (pour améliorer la visibilité sur la courbe, en pratique nous nous arrêtons quand la vraisemblance converge). Pour chaque test, nous ne retenons que le modèle optimisant le critère ICL. L'évolution de la partition est décrite par la figure 5.12, où seuls les modèles retenus sont présentés. Chaque figure est associée à une phase présentée sur la courbe du critère ICL (figure 5.11(bas)). Á chaque étape, deux composantes sont fusionnées, la dernière partition ne comprenant plus que deux composantes, au lieu de 5 pour la partition initiale. L'algorithme d'optimisation a ainsi réussi à éviter plusieurs minima locaux du critère ICL. La partition 5.12(c) obtenue à la fin de la mise à jour présente une classification plus satisfaisante que la partition initiale. Nous observons ici deux groupes de données bien distincts.

5.4.3 Collection réelle de Guillaume B.

Nous proposons d'appliquer notre algorithme d'optimisation sur une collection réelle personnelle composée de 721 images prises sur une durée de trois ans. L'utilisateur a pris les images essentiellement en France, mais aussi au Canada, en Turquie et aux USA. Les métadonnées temporelles étaient directement incluses par l'appareil photographique dans chaque image (métadonnées EXIF) et le lieu a été rajouté manuellement en fonction des coordonnées réelles. Les figures 5.6(a) et (b) page 94 présentent respectivement les métadonnées temporelles et spatiales.

Classification temporelle

La figure 5.13(a) page 102 présente la classification temporelle obtenue. Les figures 5.13(b), (c) et (d) sont des zooms sur la classification globale. La classification obtenue contient 41 composantes et les limites entre les composantes semblent généralement visuellement justifiées. Néanmoins, comme précédemment, nous observons certaines zones temporelles sur-segmentées : par exemple, les composantes sur la gauche de la figure 5.13(c). La première composante de la figure 5.13(b) est très large et nous pouvons distinguer plusieurs groupes de données non mis en valeur. Une explication peut être la structure des données qui ressemble à une distribution gaussienne : les données sont concentrées sur le centre de la classe et très étalées sur les bords.

Stabilité des classifications

Plusieurs états de la classification temporelle au cours du processus de classification sont présentés par la figure 5.14, page 103. Les modèles obtenus semblent corrects malgré quelques cas de sur-segmentation et les classes sont plus détaillées que dans le modèle final. Par exemple la première composante de ce dernier modèle (figure 5.13(b)) était initialement plus détaillée, comme on peut le voir sur les figures 5.14(a) et (b). L'algorithme a ainsi permis plusieurs fusions de composantes, ce qui semble pertinent au vu de la structure globale des données.

Classification spatiale

La classification spatiale obtenue est détaillée par la figure 5.15 page 104 : la figure (a) représente l'ensemble des données et les figures (b), (c) et (d) sont des zooms sur cette classification. Les données présentent des caractéristiques particulières : l'utilisateur a pris beaucoup d'images dans des lieux similaires, ce qui entraîne des paquets de données avec des coordonnées géographiques identiques. Les classes obtenues reflètent cette distribution des données. La majorité d'entres elles est associée à des données concentrées sur un seul point. Nous obtenons ainsi 29 composantes distinctes, chacune représentant un lieu précis de la collection.

Certaines parties de l'espace comprennent beaucoup de composantes très regroupées par rapport à l'échelle globale. Notre algorithme n'a pas réussi à les fusionner. La structure des données, concentrées en un seul point, entraîne une vraisemblance élevée pour ces composantes et une fusion de ce type de classe n'améliore pas le critère ICL. Une gaussienne représente mal un ensemble de données divisé en deux sous-groupes, chacun concentré en deux points distincts.

Stabilité des classifications

Les différents modèles obtenus, présentés dans la figure 5.16 page 105, restent similaires à ceux obtenus dans le modèle global. Nous avons noté que peu de fusions ou divisions de composantes étaient

réalisées au cours des mises à jour. Ceci s'explique par la structure des données.

Dans ces deux dernières expériences, nous avons vérifié que les partitions, temporelles et spatiales, restent stables dans le temps. Peu de données entraînent un nombre important de modifications. Ce point est important pour un utilisateur qui veut pouvoir parcourir facilement sa collection, trop de changement dans les partitions pouvant entraîner de la confusion.

5.5 Conclusion

Nous avons présenté dans cette partie un algorithme incrémental d'optimisation. Les données sont ajoutées une à une dans le modèle et la partition est mise à jour en optimisant le critère ICL à l'aide de fusions et divisions de composantes. Notre proposition est ainsi automatique puisque les classifications évoluent en fonction de la structure des données. Les expériences ont de plus montré que l'aspect incrémental est bien géré. Les partitions restent stables dans le temps et évoluent convenablement.

Parcourir sur un appareil mobile des partitions obtenues avec cette technique peut néanmoins présenter un problème quand le nombre de composantes devient important, ou quand des composantes sont très générales, comprenant un trop grand nombre d'images. Le chapitre suivant propose de combiner cet algorithme avec un algorithme agglomératif pour obtenir des partitions hiérarchiques.

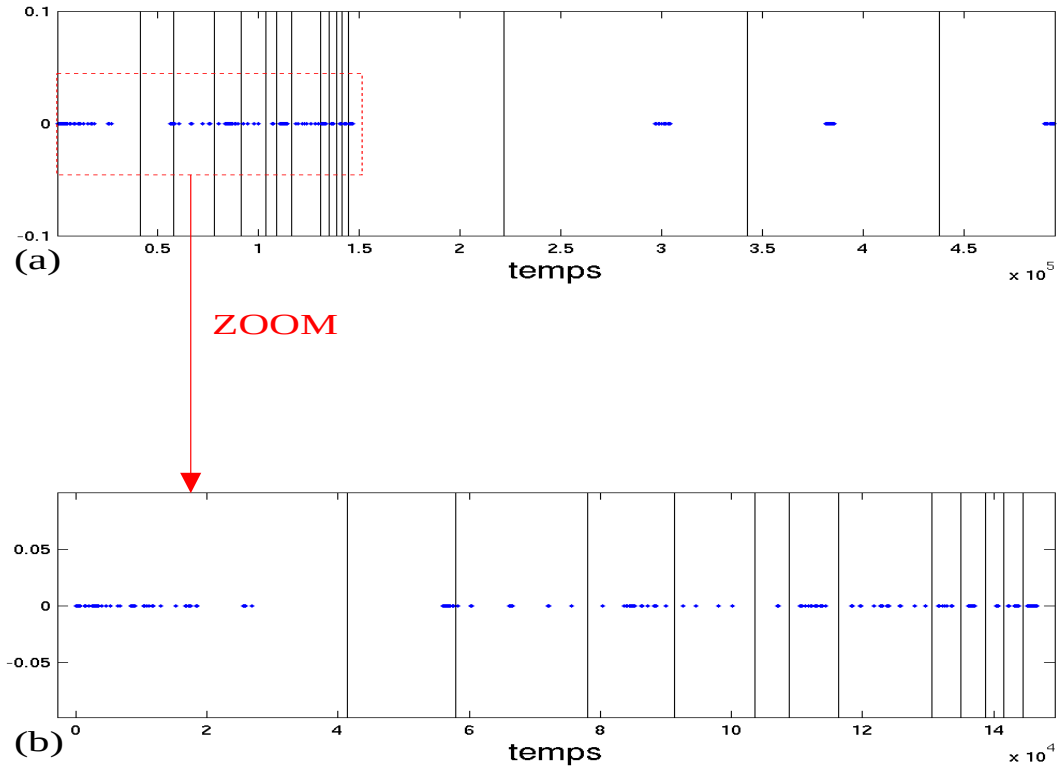


Figure 5.7: Collection artificielle réaliste : classification temporelle obtenue avec notre algorithme d'optimisation. La figure (a) représente la partition globale. La partie sélectionnée par un rectangle en pointillés est détaillée par la figure (b). Les lignes représentent les limites entre les composantes selon le critère MAP et les points les métadonnées temporelles. Les groupes semblent correctement retrouvés, les limites entre les classes étant pour la majorité visuellement justifiées.

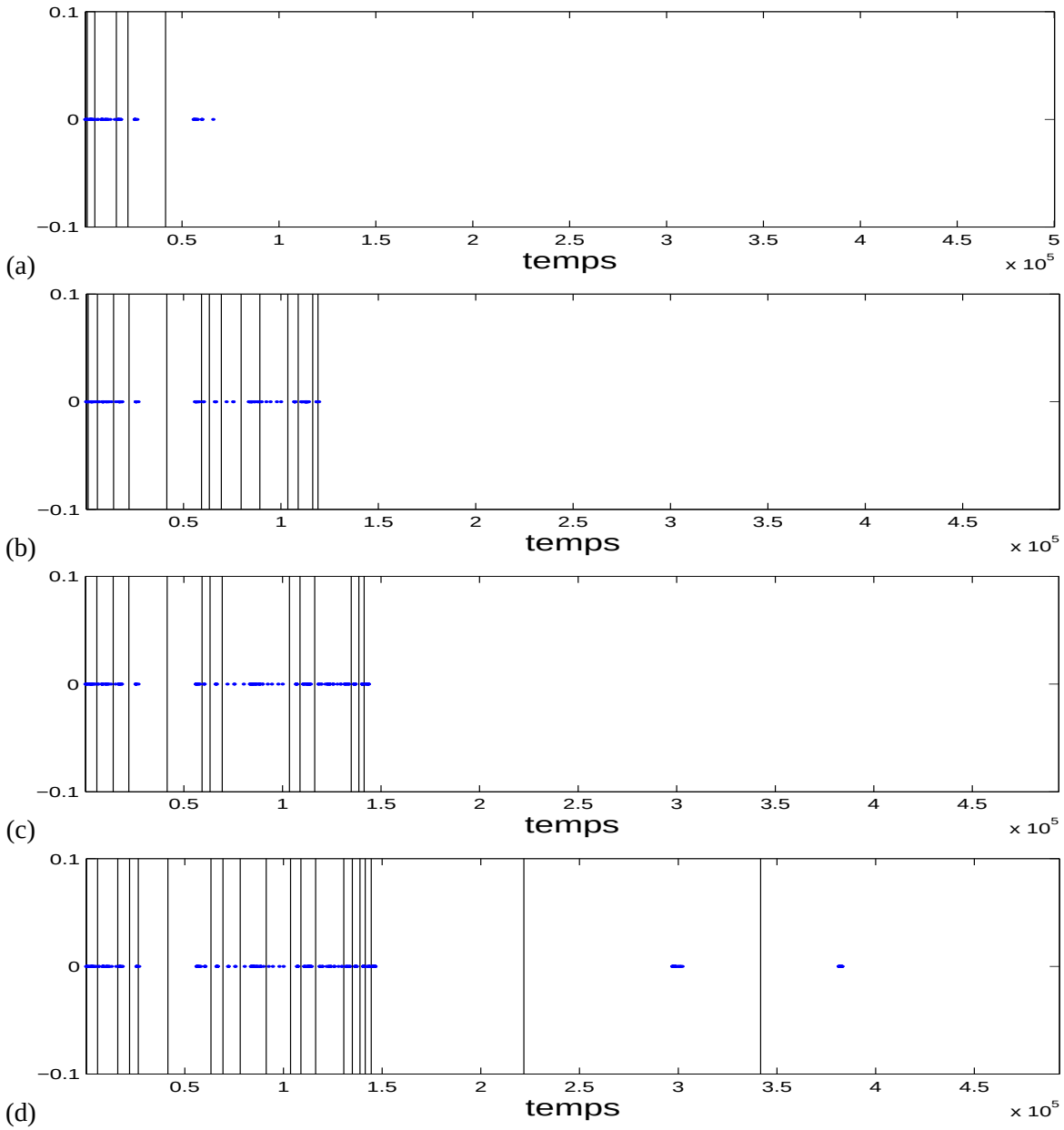


Figure 5.8: Collection artificielle réaliste : classification temporelle. Différents états de la classification obtenus au cours du processus de classification, pour 90 *(a),* 180 *(b),* 270 *(c) et* 360 *(d) données. La structure des données est correctement mise en valeur dans les figures. Notons que les données associées à la première composante obtenue dans le modèle final (la première composante à partir de la gauche sur 5.7(b)) étaient initialement plus détaillées.*

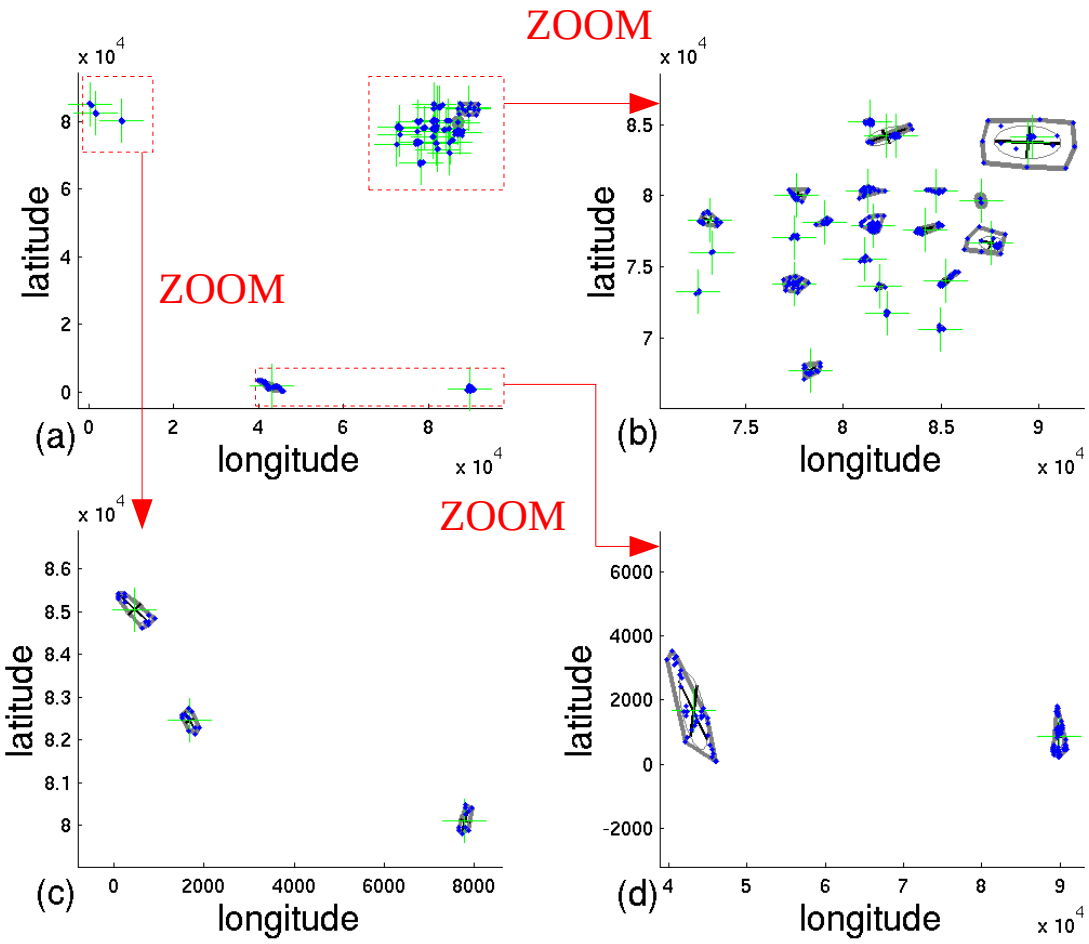


Figure 5.9: Collection artificielle réaliste : classification spatiale. Les points représentent les métadonnées spatiales, et les "+" et les ellipses indiquent respectivement les centres et les variances des composantes. Les polygones englobants représentent l'association des données aux composantes selon le critère MAP. Les figures (b), (c) et (d) sont des zooms sur la classification, définis par les carrés en pointillés.

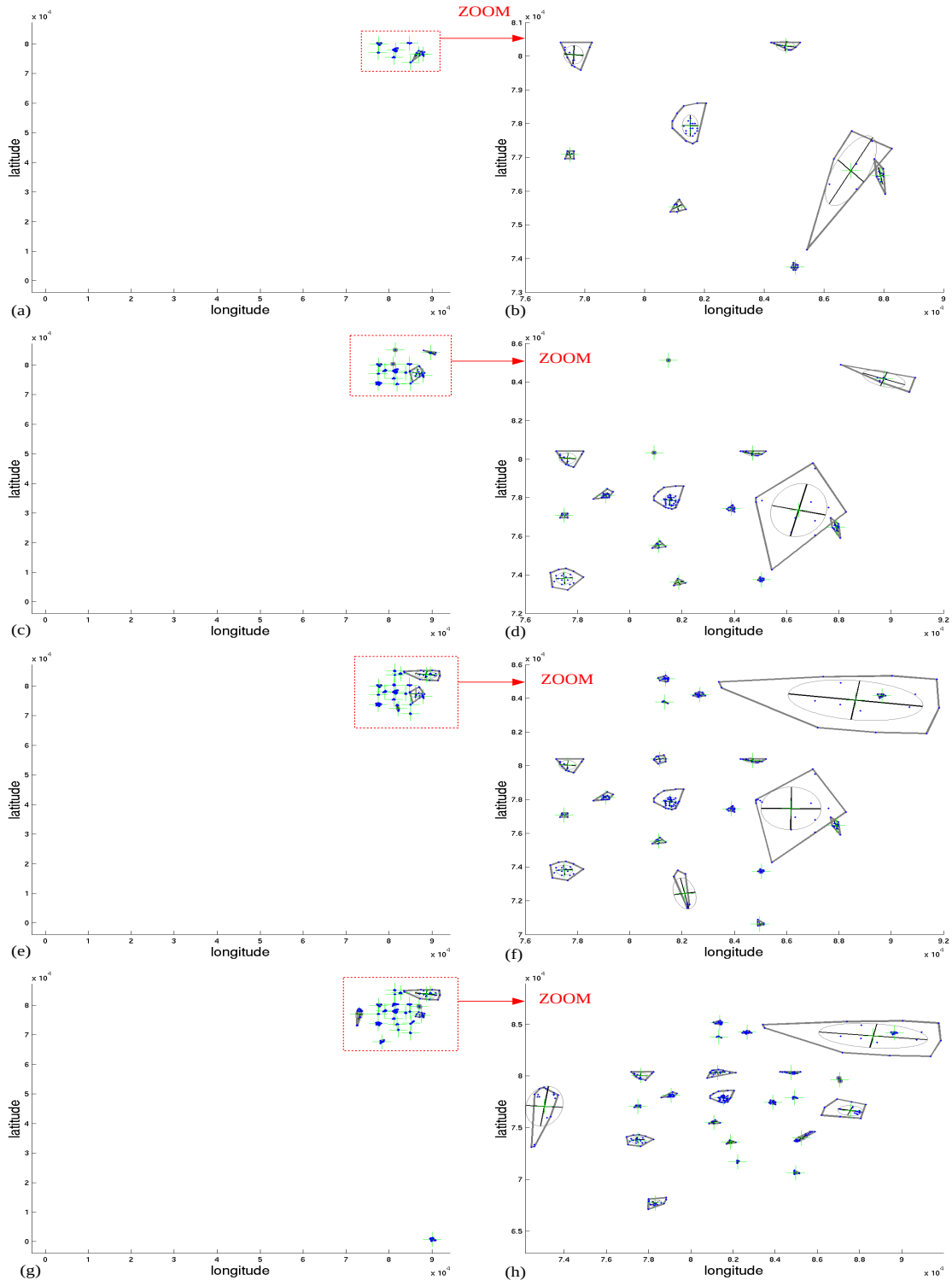


Figure 5.10: Collection artificielle réaliste : classification spatiale obtenue après 90 *(a),* 180 *(c),* 270 *(e),* 360 *(g) images. Les figures (b), (d), (f) et (h) sont respectivement des zooms sur les figures (a), (c), (e) et (g).*

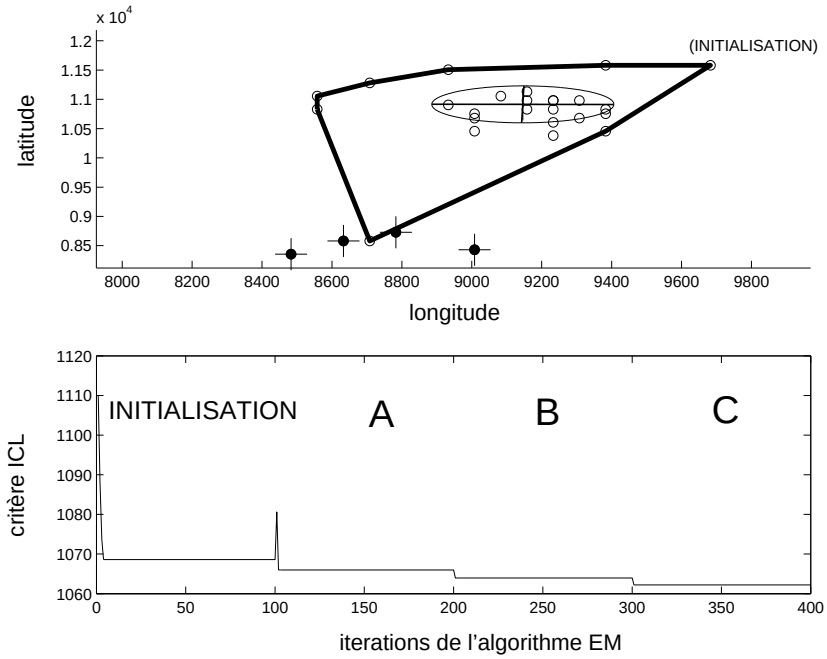


Figure 5.11: Collection artificielle réaliste : (en haut) le modèle initial optimisé avec l'algorithme EM après l'ajout d'une nouvelle donnée, et (en bas) la variation du critère ICL pendant les phases du processus d'optimisation (tests de fusions et divisions de composantes). La courbe ne présente que les optimisations obtenues quand le critère ICL est amélioré (et donc quand la modification sur le modèle est retenue). Les partitions retenues sont présentées par la figure 5.12, avec la lettre correspondante.

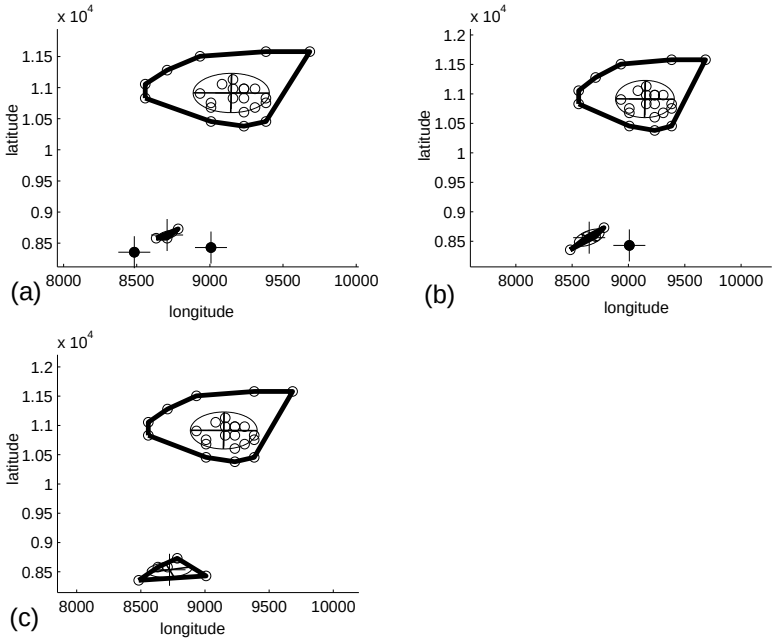


Figure 5.12: Collection artificielle réaliste : (a), (b) et (c) représentent les partitions retenues après des fusions successives de composantes. La variation de leur critère ICL après chaque modification est représentée par la figure 5.11(en bas), par la lettre correspondante. La partition (c) est suivie de plusieurs tests de fusions et divisions de composantes n'améliorant pas le critère ICL et est ainsi retenue. Elle présente un résultat plus pertinent que la partition initiale.

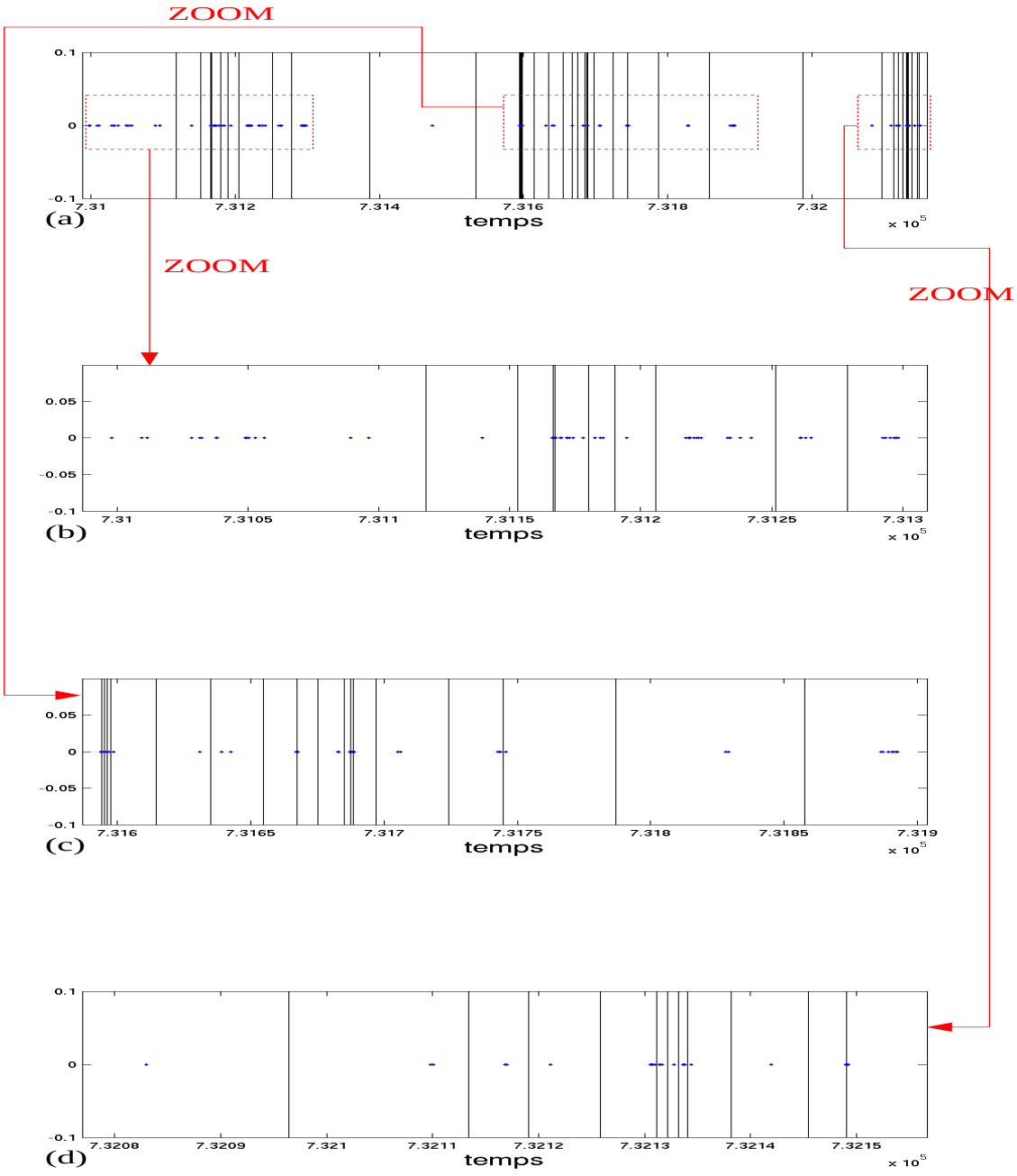


Figure 5.13: Collection réelle : classification temporelle. La classification globale obtenue est présentée par la figure (a). Les figures (b), (c) et (d) sont des zooms sur cette classification, définis par les carrés en pointillés. Les lignes représentent les limites entre les composantes et les points les métadonnées temporelles.

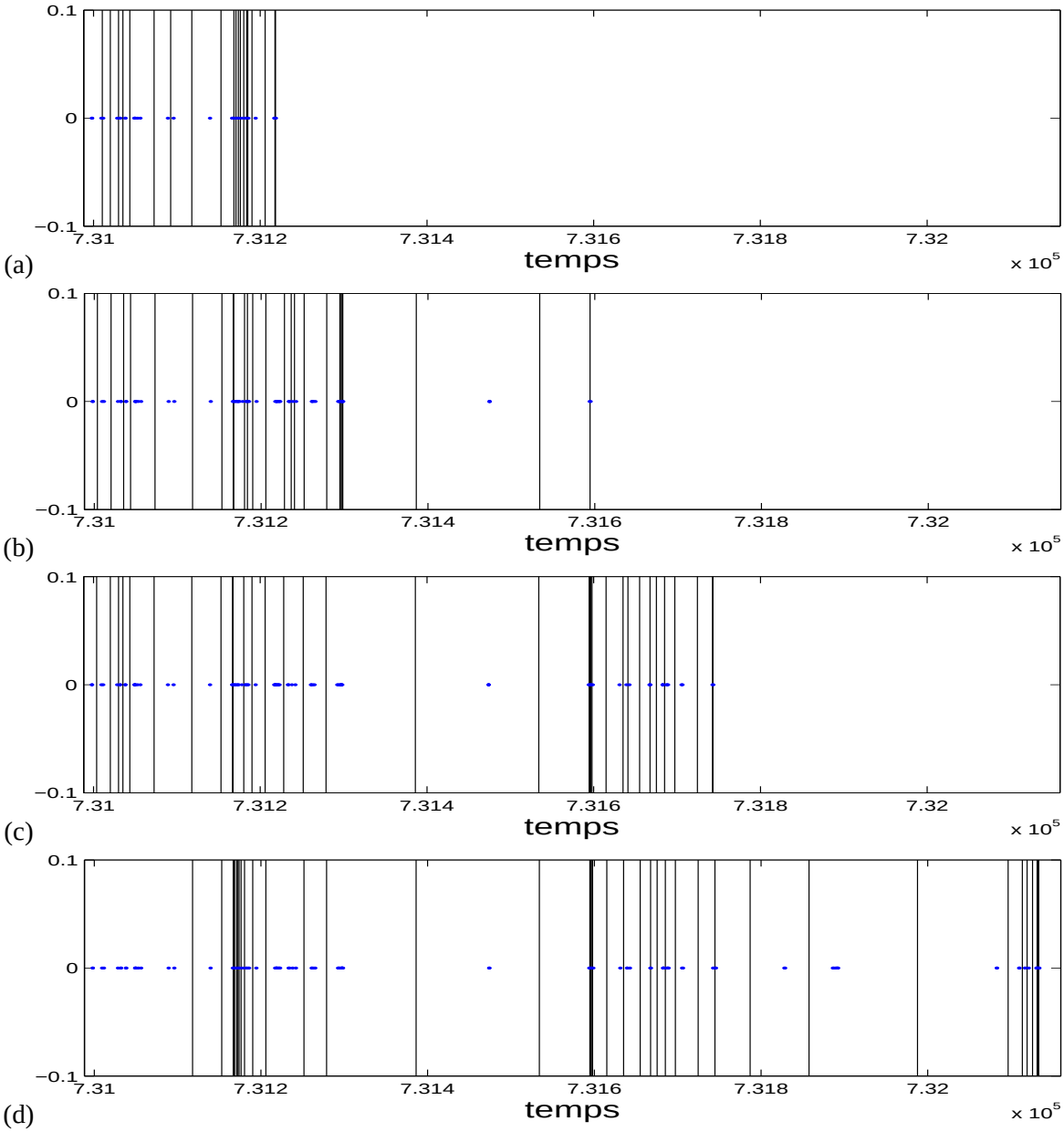


Figure 5.14: Collection réelle : classification temporelle. Différents états de la classification obtenus au cours du processus de classification, pour 150 *(a),* 300 *(b),* 450 *(c) et* 600 *(d) données.*

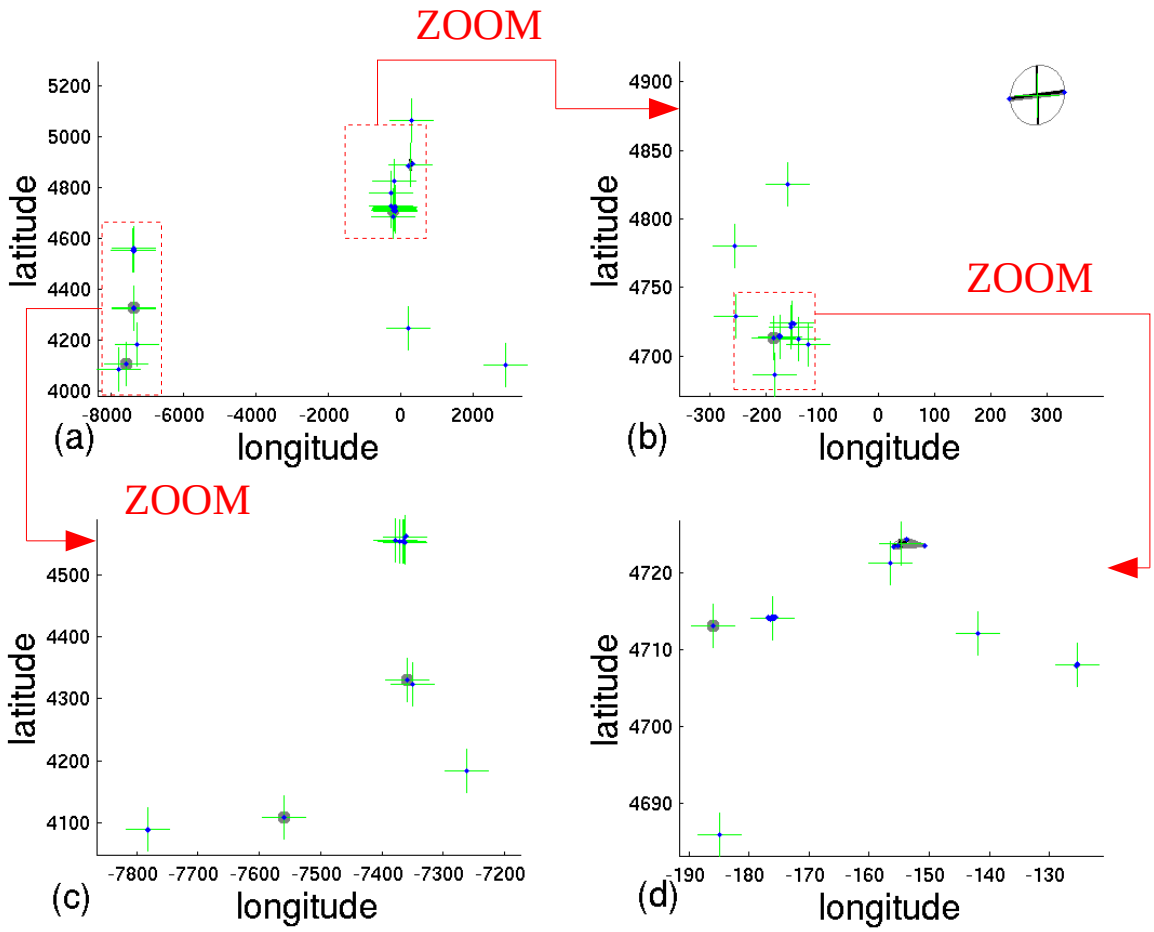


Figure 5.15: Collection réelle : classification spatiale. Les points représentent les métadonnées spatiales, les "+" et les ellipses indiquent respectivement les centres et les variances des composantes. Les polygones englobants représentent l'association des données aux composantes selon le critère MAP. Les figures (b), (c) et (d) sont des zooms sur la classification, définis par les carrés en pointillés.

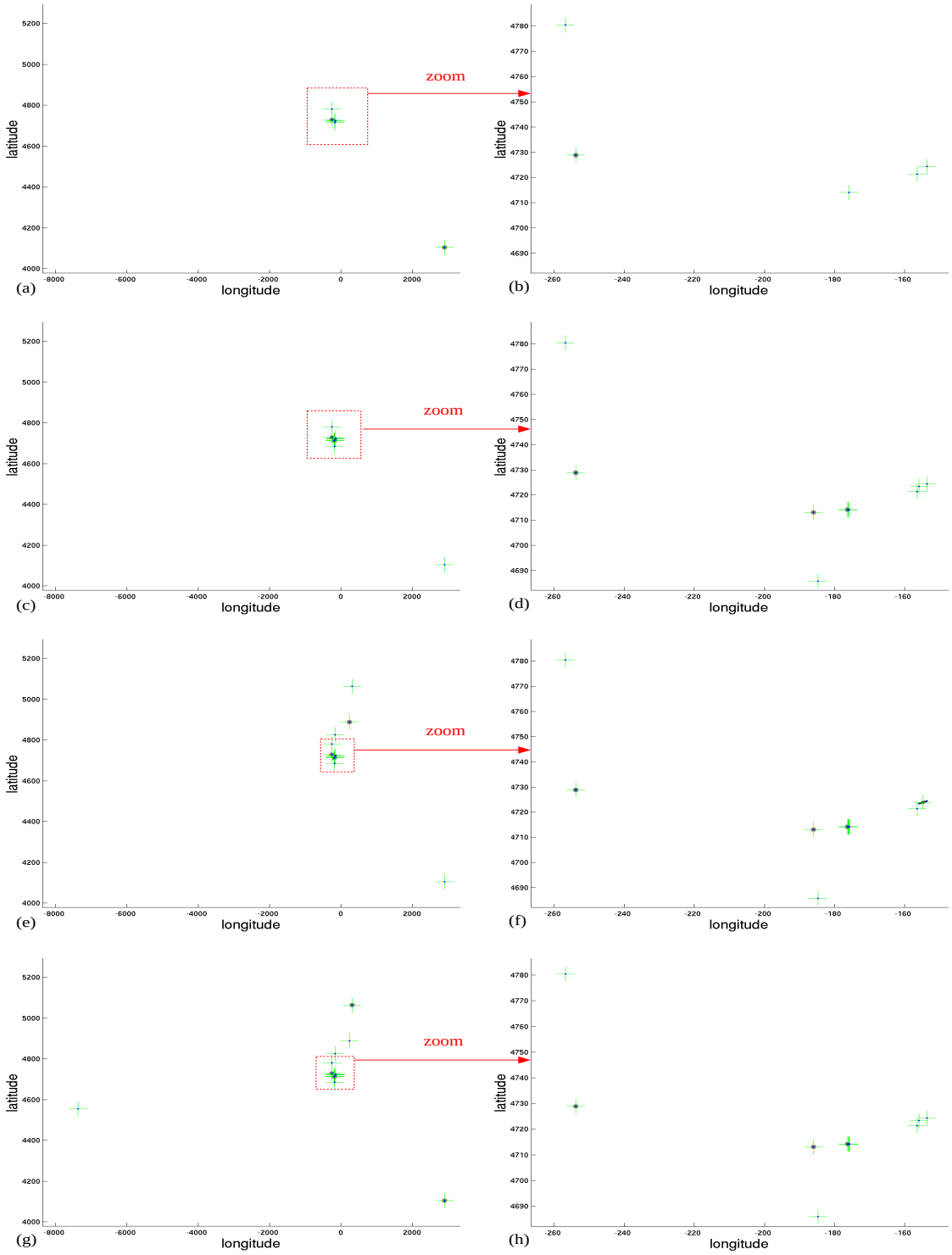


Figure 5.16: Collection réelle : classification spatiale. Différents états de la classification obtenus au cours du processus de classification, pour 150 *(a),* 300 *(c),* 450 *(e) et* 600 *(g) données. Les figures (b), (d), (f) et (h) sont respectivement des zooms sur les figures (a), (c), (e) et (g).*

CHAPITRE 6

Structuration hiérarchique incrémentale de la collection

Nous présentons dans ce chapitre notre algorithme incrémental hiérarchique. Il est obtenu en combinant notre procédure d'optimisation présentée au chapitre précédent, un algorithme de sélection de niveaux proposé dans ce chapitre, et une approche agglomérative [44].

6.1 Introduction

L'objectif est de fournir en sortie, non plus une partition, mais une classification hiérarchique de la collection. Nous rappelons que ce point est important pour développer une IHM pertinente sur un appareil mobile. Une hiérarchie de classes permet de disposer de résumés des différents événements de la collection et facilite son parcours. Nous proposons dans ce chapitre un algorithme hiérarchique incrémental combinant une procédure agglomérative [44] et notre algorithme d'optimisation du critère ICL (voir chapitre précédent).

De nombreux algorithmes hiérarchiques existent dans la littérature [8, 62], mais aucun ne semble pertinent pour notre cas d'utilisation. Nous nous sommes restreints à l'étude des algorithmes permettant de classer des données numériques. Ces algorithmes sont généralement développés pour pouvoir résumer de larges bases de données et doivent prendre en compte des contraintes ne nous concernant pas (la taille des données par exemple qui est plus limitée dans notre cas). Nous n'avons de plus trouvé aucun algorithme hiérarchique incrémental. Bien que GRIN [31] et Birch [125] soient dits "incrémentaux", ils doivent connaître l'ensemble des données à classer pour fonctionner : ils ne sont donc pas incrémentaux dans le sens où nous l'entendons. Ils dépendent de plus de paramètres critiques à régler manuellement.

Étant dans un contexte de modèles de mélange gaussien, nous avons choisi de combiner notre algorithme d'optimisation avec un algorithme hiérarchique basé sur les modèles gaussiens. Nous gardons ainsi tous les avantages liés à ce type d'approche, en particulier l'évaluation de la qualité des partitions grâce aux critères statistiques. Ce point est important pour pouvoir évaluer les partitions de la classification et ainsi faire une sélection des meilleurs niveaux à présenter à l'utilisateur.

Nous présentons tout d'abord les différents algorithmes hiérarchiques basés sur les modèles gaussiens existants dans la littérature. Nous proposons ensuite une approche à partir de l'algorithme de Fraley [44], permettant d'obtenir un arbre binaire, dans lequel nous sélectionnons seulement les niveaux de bonne qualité. Enfin nous présentons notre algorithme hiérarchique incrémental.

6.2 État de l'art sur les algorithmes hiérarchiques basés sur les modèles de mélange gaussien

Nous présentons dans cette partie trois approches basées sur les modèles de mélange. La première méthode est une approche dépendante de paramètres arbitraires, mais pertinente pour obtenir rapidement une partition dans le cas où l'on dispose de beaucoup de données. La deuxième méthode propose un algorithme itératif et un critère d'erreur, afin de résumer un modèle de mélange. Enfin, la troisième approche est plus classique puisqu'elle propose un algorithme hiérarchique agglomératif prenant en entrée un modèle de mélange gaussien.

6.2.1 Approche par fraction

Tantrum et al. [110] ont proposé une méthode pour construire une classification hiérarchique basée sur les modèles de mélange, pour classer de larges ensembles de données.

La méthode par fraction est une approche par division proposée dans [34]. Elle consiste à diviser l'ensemble des données et à construire des partitions locales sur chaque ensemble. Les différentes étapes sont :

1. diviser les observations en plusieurs fractions de taille M ;

2. classer chaque fraction en ρM classes, avec $\rho < 1$. Résumer chaque classe avec son centre. Les centres obtenus sont appelés méta-observations ;

3. si le nombre de méta-observations est supérieur à M, retourner à l'étape 1. Les méta-observations prennent la place des observations initiales ;

4. classer les méta-observations en G classes ;

5. Assigner chaque observation initiale à la classe ayant son centre le plus proche.

Deux problèmes sont liés à cette approche. Tout d'abord, elle dépend de paramètres critiques : les paramètres M et G doivent être fournis par l'utilisateur. Ensuite les fractions créées à l'étape 1 ne peuvent pas être remises en cause. Une fois que les observations d'un même groupe sont affectées à des méta-observations différentes, l'erreur ne peut pas être corrigée.

Tantrum et al. [110] ont adapté cette méthode à l'approche par modèle et proposé une amélioration pour corriger le défaut précédent. Une méta-observation est maintenant définie par son centre, sa variance et le nombre d'observations associées. Ils proposent d'itérer plusieurs fois les différentes étapes $(1-4)$ et de former les fractions d'observations de l'étape 1, à l'itération i, en se basant sur la classification obtenue à l'itération $(i-1)$. L'étape 4 est modifiée comme suit :

4.a. classer les méta informations en G classes, où G est déterminé à l'aide d'un critère statistique ;

4.b. redéfinir les fractions de l'étape 1 à l'itération $i-1$: appliquer un algorithme hiérarchique agglomératif avec les observations initiales. Dès qu'une composante contient plus de M observations, on crée une fraction avec ses observations et on enlève la classe du processus de fusion.

L'algorithme s'arrête quand les classifications obtenues présentent peu de différences. Cette nouvelle étape permet ainsi de remettre en cause les fractions de données réalisées à l'étape 1, en prenant plus en compte la structure des données.

Cette technique est néanmoins basée sur des paramètres critiques fixés manuellement et semble plus appropriée pour classer un grand nombre d'observations rapidement. Le paramètre M est un paramètre arbitraire défini en fonction du nombre de données, et pour l'étape 4.a, l'utilisateur doit fournir la complexité des modèles à tester. La topologie de l'arbre est fixée par l'ensemble de ces paramètres alors qu'elle ne devrait dépendre que de la structure des données. L'algorithme semble peu approprié pour notre cas d'utilisation.

6.2.2 Résumé d'un modèle de mélange

Goldberger et Roweis [53] ont proposé un algorithme permettant de résumer un modèle de mélange. L'algorithme consiste à regrouper les composantes d'un modèle initial en un nombre inférieur de composantes. L'utilisateur doit fournir le nombre de composantes à obtenir en sortie et l'algorithme recherche ensuite le modèle le plus similaire au modèle initial, comprenant le nombre demandé de composantes. Le problème est de déterminer quelles composantes doivent être regroupées ensembles pour présenter un bon résumé du modèle.

La proposition est un algorithme itératif en deux phases : la première consiste à regrouper les composantes du modèle initial à partir d'une fonction de transformation et la seconde calcule le critère d'erreur, basé sur une divergence modifiée de Kullback-Leibler. L'objectif est de minimiser ce critère.

Soit f le modèle initial de K composantes et g un modèle de sortie de K' composantes définis par :

$$f(x) = \sum_{k=1}^{K} p_k \cdot \mathcal{N}(x, \theta_k) = \sum_{k=1}^{K} p_k \cdot f_k(x), \tag{6.1}$$

$$g(x) = \sum_{k=1}^{K} p_k \cdot g_k(x), \tag{6.2}$$

La distance entre deux modèles est définie par :

$$dist(f, g) = \sum_{k=1}^{K} p_k \cdot \min_{k'=1}^{K'} KL(f_k || g_{k'}), \tag{6.3}$$

où KL est une divergence de Kullback-Leibler modifiée entre uniquement deux gaussiennes. Elle est définie par :

$$KL(\Sigma_i, \Sigma_j) = \frac{1}{2}(log\frac{|\Sigma_j|}{|\Sigma_i|} + tr(\Sigma_j^{-1}\Sigma_i) + (\mu_i - \mu_j)^t\Sigma_j^{-1}(\mu_i - \mu_j) - d), \tag{6.4}$$

où d est la dimension des données.

La distance $dist(f, g)$ est le coût pour approximer le modèle f en un modèle g comprenant un nombre inférieur de composantes. Pour un modèle $g \in H_{K'}$, on définit une fonction de transformation $\pi^g \in S$ par :

$$\pi^g(k) = arg \min_{k'=1}^{K'} KL(f_k||g_{k'}),\ i = 1, \ldots, K \tag{6.5}$$

Ainsi la distance $dist(f, g)$ devient :

$$dist(f, g) = dist(f, g, \pi^g) = \min_{\pi \in S} dist(f, g, \pi) \tag{6.6}$$

Le terme π^g est donc la transformation associant une composante de f vers g présentant la plus petite distance entre les deux modèles.

Le modèle réduit optimum est une solution de la double minimisation suivante :

$$\hat{g} = arg \min_{g} \min_{\pi \in S} dist(f, g, \pi) \tag{6.7}$$

Aucun algorithme exact n'existe pour trouver une solution optimale globale mais les auteurs proposent un algorithme itératif, permettant d'obtenir un minimum local. Soit une transformation $\pi \in S$, $g^\pi \in H_{K'}$ est défini comme suit. Pour chaque k' tel que $\pi^{-1}(k')$ est non vide, on définit la densité suivante :

$$f_{k'}^\pi = \frac{\sum_{k \in \pi^{-1}(k')} p_k \cdot f_k}{\sum_{k \in \pi^{-1}(k')} p_k} \tag{6.8}$$

Le centre et la variance de l'ensemble $f_{k'}^\pi$ sont définis par :

$$\mu'_k = \frac{1}{\beta_{k'}} \sum_{k \in \pi^{-1}(k')} p_k \cdot \mu_k, \quad \Sigma'_k = \frac{1}{\beta_{k'}} \sum_{k \in \pi^{-1}(k')} p_k \cdot (\Sigma_k + (\mu_k - \mu'_k)(\mu_i - \mu'_k)^t), \tag{6.9}$$

où $\beta_{k'} = \sum_{k \in \pi^{-1}(k')} p_k$. Soit $g_k^\pi = (\mu'_k, \Sigma'_k)$ la composante gaussienne obtenue en fusionnant l'ensemble de composantes $f_{k'}^\pi$. Cela satisfait :

$$g_{k'}^\pi = arg \min_{g} KL(f_{k'}^\pi||g) = arg \min_{g} dist(f_k^\pi, g) \tag{6.10}$$

Ainsi la version fusionnée de f suivant la transformation π est définie par :

$$g^\pi = \sum_{k'=1}^{K'} \beta_{k'} \cdot g_{k'}^\pi \tag{6.11}$$

L'algorithme consiste à tester plusieurs fonctions $\pi \in S$ et à retenir le modèle g^π associé, minimisant la distance entre f et g. L'algorithme est divisé en deux phases :

1. **regroupement** :
$$\pi^g = arg \min_{\pi} dist(f, g, \pi),$$
où les paramètres du modèle g sont calculés à partir de la transformation π ;
2. **optimisation** : la distance entre f et g est calculée,
$$g^\pi = arg \min_{g} dist(f, g, \pi).$$

Pour trouver un bon minimum du critère d'erreur, il est nécessaire de tester plusieurs transformations entre f et g. Et à chaque itération de l'algorithme, on garde le modèle g minimisant une divergence de Kullback-Leibler modifiée.

La construction de la hiérarchie est donc réalisée en se basant seulement sur les paramètres du modèle de mélange, ce qui présente un avantage du point de vue de la complexité de l'algorithme. Cet algorithme présente néanmoins plusieurs problèmes :

- il est dépendant de la fonction de transformation π. En pratique, il est nécessaire de la générer aléatoirement. L'algorithme permet de trouver seulement un minimum local puisque on ne peut pas tester toutes les solutions de transformation. Le résultat fourni avec une telle approche n'est donc pas déterministe, ce qui peut poser problème pour une classification d'images personnelles. Il est nécessaire d'avoir un résultat stable au fur et à mesure de l'ajout de nouvelles données ;

- il est nécessaire de fournir la complexité du modèle de sortie g.

Néanmoins une perspective de travail est d'adapter cet algorithme avec notre algorithme d'optimisation pour fournir des classifications hiérarchiques. Des expériences ont montré que les résultats pour résumer un modèle détaillé étaient bons, mais il nous reste encore à trouver le moyen pour le combiner avec notre algorithme incrémental d'optimisation du critère ICL.

6.2.3 Approche par minimum de variance

Cet algorithme est plus classique que les deux algorithmes précédents. Il est basé sur une approche par minimum de variance et a été adapté sur les modèles de mélange gaussien par Fraley [44]. Il propose un algorithme agglomératif à partir d'un modèle de mélange et définit les critères d'erreur à optimiser pour chaque type de contraintes sur les modèles.

Algorithme de variance minimum

Les algorithmes de variance minimum sont basés sur une matrice de distance des observations deux à deux, appelée matrice de connectivité. L'optimisation d'une fonction dépendant de cette matrice permet ensuite de construire la classification. Un critère utilisant une telle matrice est le critère d'erreur quadratique popularisé par l'analyse de variance et d'autres procédures statistiques. Il est aussi utilisé dans des algorithmes de classifications, par exemple, l'algorithme k-means. Pour une classe k, il est défini par :

$$W_k = \sum_{i=1}^{n_k} (x_i - \mu_k)^2 \tag{6.12}$$

Ainsi pour un ensemble de classes, le critère de Ward [121] est défini par :

$$W = \sum_{k=1}^{K} W_k \tag{6.13}$$

La méthode de Ward fusionne à chaque itération les deux classes minimisant ce dernier critère. Cela revient à minimiser le critère $\triangle W_{\langle i,j \rangle}$ définit par la variation de W, obtenu après une fusion de deux composantes i et j. Le critère W augmente au fur et à mesure des fusions mais cette augmentation doit être minimale.

$\Sigma_k = \lambda B$	$tr\left(\sum_{k=1}^{K} W_k\right) = \sum_{k=1}^{G}$
$\Sigma_k = \lambda_k B_k$	$\sum_{k=1}^{K} n_k log tr\left(\frac{W_k}{n_k}\right) = \sum_{k=1}^{K} log\left[\frac{tr(W_k)}{n_k}\right]$
$\Sigma_k = \lambda C$	$\sum_{k=1}^{K} \lvert W_k \rvert$
$\Sigma_k = \lambda_k C_k$	$\sum_{k=1}^{K} n_k log\left\lvert\frac{W_k}{n_k}\right\rvert$

Table 6.1: Critères d'optimisation de l'algorithme de Fraley pour différentes contraintes sur les matrices de variance.

Algorithme 3 Algorithme hiérarchique agglomératif de Fraley

initialisation : un modèle de mélange gaussien de K composantes ;
1.calcul de la matrice de connectivité M à partir du critère d'erreur W et de l'équation 6.15;
tantque le nombre de composantes est supérieur à 1 **faire**
2.1.sélectionner les composantes i et j à fusionner minimisant le critère 6.14 ;
2.2.mise à jour de la matrice de connectivité à l'aide de l'équation 6.15.
fin tantque

Adaptation au modèle de mélange gaussien

Fraley [44] propose d'adapter cet algorithme aux modèles de mélange gaussien. La seule différence avec l'algorithme précédent réside dans le choix du critère à optimiser. Selon le choix des contraintes à priori sur le modèle, les critères sont différents. Le tableau 6.1 ci-dessus résume les différents critères à minimiser pour 4 modèles classiques. Dans notre cas, le critère à minimiser est :

$$\sum_{k=1}^{K} n_k \cdot log\Big|\frac{W_k}{n_k}\Big|, \tag{6.14}$$

puisque nous avons choisi des contraintes à priori sur le modèle de mélange telles que les matrices de variance soient de la forme $\Sigma_k = \lambda_k C_k$.

La matrice de connectivité est définie par les termes $W_{\langle ij \rangle}$ ($1 \leq i \leq K$ et $i \leq j \leq K$) représentant le coût de la fusion entre les composantes i et j. Ce coût peut être obtenu directement à partir de W_i et W_j :

$$W_{\langle i,j \rangle} = W_i - W_j + w_{ij} w'_{ij}, \tag{6.15}$$

où

$$w_{ij} = \eta_{ji} s_i - \eta_{ij} s_j, \eta_{ij} = \sqrt{\frac{n_i}{n_j(n_i + n_j)}}, \tag{6.16}$$

où s_k est la somme des observations de la composante k. Cette récurrence permet d'obtenir un algorithme basé uniquement sur les paramètres des modèles de mélange et réduit sa complexité. A chaque itération, seule une mise à jour de la matrice de connectivité est nécessaire pour chaque fusion de composantes réalisée. La complexité en $O(K^2)$ reste raisonnable puisque dans notre contexte nos partitions contiennent un nombre limité de composantes. Dans le pire des cas, nous appellerons cet algorithme sur l'ensemble des feuilles de notre arbre.

Nous allons donc appliquer cet algorithme sur des partitions obtenues avec notre algorithme d'optimisation. L'algorithme 3 ci-dessus explicite en détail la technique de construction. L'approche par modèle de mélange fournit un avantage sur les autres approches puisque chaque niveau est un modèle de

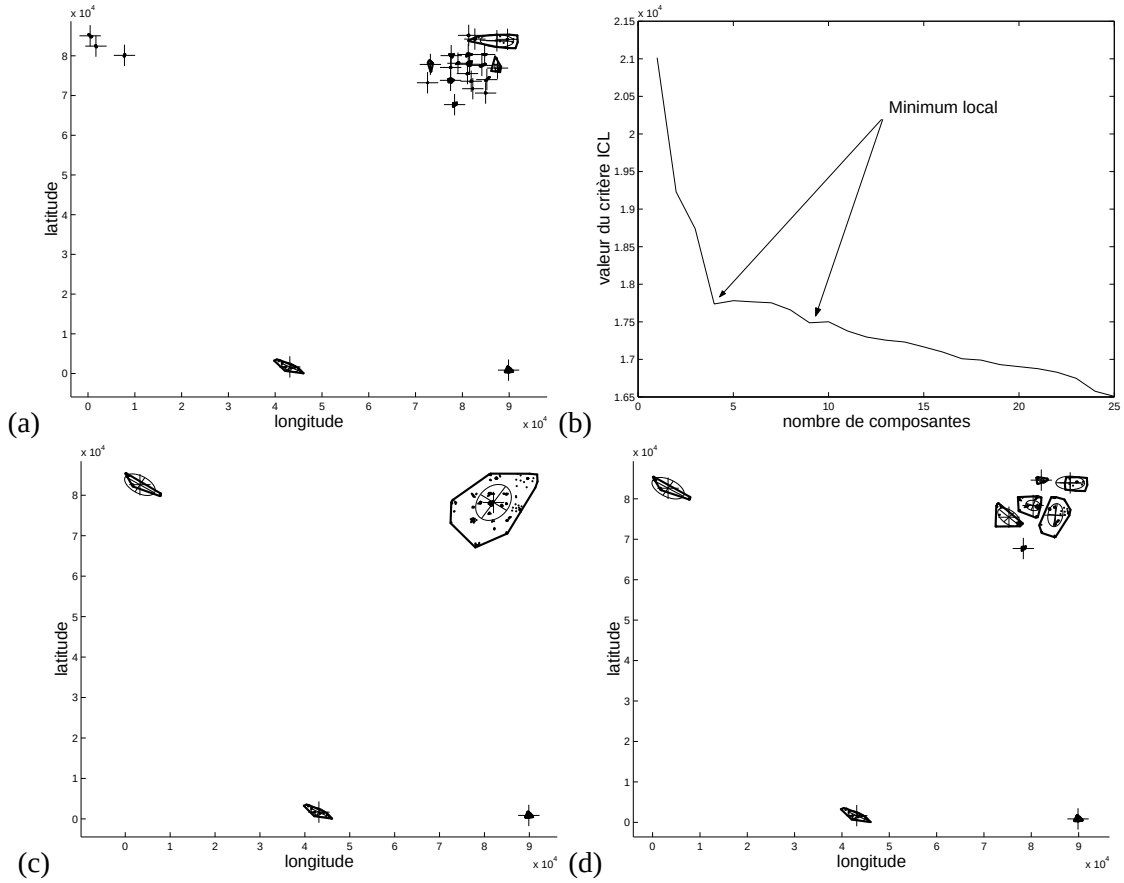


Figure 6.1: Exemple de sélection de niveaux à l'aide des minima du critère ICL. La figure (a) est la partition initiale des données spatiales de la collection artificielle réaliste, obtenue au chapitre précédent avec notre algorithme d'optimisation. La figure (b) représente les valeurs du critère ICL pour chaque niveau de l'arbre. Enfin les figures (c) et (d) sont des partitions associées à un minimum local du critère ICL.

mélange pouvant être évalué avec les techniques explicitées dans le chapitre 4. L'arbre fourni en sortie avec l'algorithme 3 est binaire, ce qui pose le problème de la pertinence de chaque niveau. Nous proposons dans la suite une méthode pour sélectionner les partitions pertinentes.

6.3 Sélection des niveaux d'un arbre binaire avec le critère ICL

Notre classification hiérarchique ne doit contenir que des niveaux pertinents, c'est à dire, avec des classes distinctes et des données bien regroupées pour faciliter le parcours de l'arbre par un utilisateur. L'arbre binaire fournit par l'algorithme de Fraley ne garantit pas que les niveaux présentent ces propriétés.

Algorithme 4 Recherche de partition avec un minimum local du critère ICL

en entrée : un arbre binaire construit à partir d'un modèle de mélange. Chaque niveau est un modèle de mélange gaussien.
initialisation : la variable i fait référence au niveau de l'arbre et est initialisée à $i = 1$.
$ICL(i)$ est le critère ICL associé à la partition du niveau i.
tantque i n'est pas associé aux feuilles de l'arbre **faire**
1.calculer le critère $ICL(i+1)$;
si $ICL(i) < ICL(i+1)$ **alors**
2.1.sélectionner le niveau i ;
pour chaque noeud q de la partition du niveau i **faire**
2.2.appeller récursivement la fonction pour le sous-arbre de racine q ;
fin pour
2.3.aller à l'étape 5 ;
finsi
3.$i = i + 1$;
fin tantque
si i est associé aux feuilles de l'arbre **alors**
4.le niveau des feuilles est sélectionné {quand aucun niveau ne présente de minimum local du critère ICL, nous sélectionnons par défaut le niveau des feuilles} ;
finsi
5.fin de la fonction.

Notre objectif est de faire une sélection des niveaux susceptibles de bien représenter la structure de la collection.

Critère de sélection

La sélection de niveaux dans une hiérarchie revient à comparer les partitions obtenues dans l'arbre. Celui-ci fournit les différentes solutions de partitions pour un nombre variable de composantes et chaque niveau est un modèle de mélange. Le fait de disposer d'une classification hiérarchique binaire présente un avantage certain : le nombre de partitions à comparer est limité, l'espace des paramètres des modèles étant déterminé pendant la construction de l'arbre.

Nous avons le choix entre deux techniques pour sélectionner les niveaux pertinents : la détection d'un coude dans une courbe d'erreur ou la comparaison avec des critères statistiques. La partition initiale étant obtenue à l'aide du critère statistique ICL et l'évaluation des partitions étant directe avec cette méthode, nous avons choisi de garder la même approche et de comparer les partitions avec ce critère. Nous rappelons que le critère ICL pénalise la vraisemblance d'un modèle par sa complexité et par un critère de classifiabilité, ce qui est pertinent pour choisir des partitions avec des composantes distinctes.

Méthode de sélection

La méthode de sélection, proposée dans [51], consiste à retrouver les niveaux de la partition présentant un minimum local du critère ICL. De tels niveaux sont censés être une représentation pertinente plus générale que la partition la plus fine de l'arbre. Notre objectif est de retrouver toutes les partitions dans l'arbre binaire présentant un minimum local du critère ICL. Nous ne présenterons que ces partitions à l'utilisateur afin de faciliter le parcours de la collection.

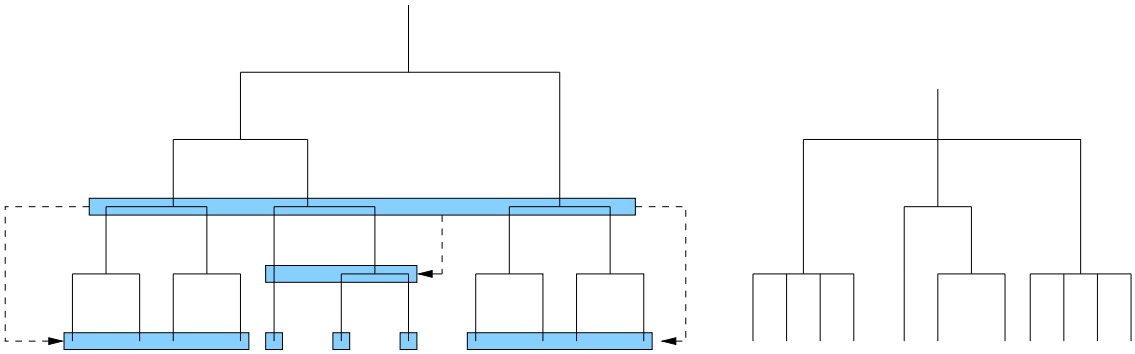


Figure 6.2: Selection de niveaux avec l'algorithme 4. La figure de droite est l'arbre initial. Les niveaux présentant des minima du critère ICL sont représentés par des rectangles gris. La figure de droite représente l'arbre obtenu après la sélection des niveaux : seuls les niveaux ayant un minimum du critère ICL apparaissent.

Nous présentons un exemple sur les métadonnées spatiales de la collection réaliste du chapitre précédent. La figure 6.1 page 113 présente un exemple de sélection de niveaux avec le critère ICL. Nous avons construit un arbre binaire obtenu avec l'algorithme de Fraley (page 112) et sélectionné les niveaux présentant un minimum local du critère ICL. La figure 6.1(a) présente la partition initiale des données et la courbe sur la figure (b) présente le critère ICL de chaque niveau de l'arbre. Nous sélectionnons tous les niveaux associés à un minimum local sur cette courbe, obtenu ici pour les modèles à 4 et 9 composantes. Les niveaux sélectionnés dans les deux cas semblent pertinents, les classes étant distinctes. Ils présentent de bonnes solutions pour représenter la structure des données.

Notre algorithme de sélection des niveaux consiste donc à parcourir l'arbre de haut en bas et à sélectionner le premier niveau présentant un minimum local du critère ICL. Une fois ce niveau obtenu, nous effectuons une recherche des minima du critère ICL dans chaque sous-arbre du niveau courant. Nous recherchons des minima locaux, et non plus globaux, du critère ICL. L'algorithme 4 ci-contre détaille cette approche et la figure 6.2 ci-dessus l'illustre par un exemple. L'arbre de gauche représente l'arbre binaire initial obtenu avec l'algorithme 3. Les niveaux présentant un minimum du critère ICL sont représentés par des rectangles gris. Nous parcourons la racine et sélectionnons le premier minimum local. Ensuite, nous lançons une recherche des minima locaux dans les sous-arbres du niveau sélectionné. Les deux autres niveaux sélectionnés sont donc des partitions d'une partie seulement des données. Les classes situées aux feuilles de l'arbre sont toujours retournées. L'arbre de droite représente la structure obtenue après la sélection des niveaux.

Propriétés de l'arbre obtenu

L'arbre obtenu est entièrement déterminé par la structure des données via l'évaluation des niveaux avec le critère ICL. Ces propriétés sont libres puisque le nombre de fils par noeud, la largeur et la profondeur dépendent du processus de sélection des minima du critère ICL. Nous proposons dans la suite un algorithme incrémental combinant cette approche avec notre algorithme d'optimisation.

6.4 Proposition d'un algorithme hiérarchique et incrémental

Notre approche est basée sur notre procédure d'optimisation, l'approche agglomérative proposée par Fraley [44], et notre algorithme de sélection de niveaux : nous les combinons pour obtenir un algorithme hiérarchique et incrémental. L'idée est de propager les nouvelles données à partir de la racine jusqu'aux feuilles, en mettant à jour chaque niveau de l'arbre et en ne réorganisant entièrement que les sous-arbres où des modifications importantes ont lieu. Avant de présenter notre algorithme, nous définissons plusieurs notions sur les noeuds des arbres.

Définitions

Définition d'un noeud

Un noeud q dans l'arbre correspond à une composante gaussienne et est défini par :

- un ensemble de donnée X_q, assigné selon les probabilités *a posteriori* maximum (à partir des variables t_{ik}) ;
- les paramètres de la composante associés à X_q.

Nous appelons c_q l'ensemble des fils de q. c_q est ainsi une partition du noeud q.

Noeud modifié et non-modifié

Un noeud q est dit *modifié* après l'ajout d'une nouvelle donnée et la mise à jour avec notre algorithme d'optimisation, si ses données X_q associées sont modifiées, sans prendre en compte la nouvelle donnée. Nous notons Q_{mod} l'ensemble des noeuds modifiés et Q_{n-mod} l'ensemble des noeuds non-modifiés.

Par exemple, si une nouvelle donnée n est ajoutée et qu'une partition initiale est définie par :

- $q_1 = \{a, b, c, d, e\}$
- $q_2 = \{f, g, h\}$
- $q_3 = \{k, l, m\}$

Supposons que nous obtenons la partition suivante après la mise à jour :

- $q_1 = \{a, b, c, d, e, n\}$
- $q'_2 = \{f, g, h, m\}$
- $q'_3 = \{k, l\}$

alors q_1 est considéré comme non-modifié alors que q_2 et q_3 le sont. q_1 n'est pas modifié puisque on ne prend pas en compte la nouvelle donnée n : nous détectons le cas où la nouvelle donnée est ajoutée à une composante sans la modifier.

Notion de généralisation et de niveau

Nous parlerons d'une *généralisation* d'une partition p s'il existe une partition p' vérifiant les propriétés suivantes :

- elle est associée à un minimum local du critère ICL ;
- elle représente le même ensemble de données que p ;
- elle est de complexité inférieure à p (la partition p' contient moins de composantes que p).

La partition p' est dite plus générale que p.

Afin de décrire les arbres obtenus, nous parlerons de *niveau* de l'arbre : cela représente un ensemble de noeuds situés à une même profondeur dans l'arbre. Nous définissons le premier niveau par la racine de l'arbre, le second niveau par les fils de la racine, ...

Mise à jour

Nous ajoutons les nouvelles données une à une par la racine de l'arbre et commençons par mettre celle-ci à jour. La procédure pour mettre à jour un noeud q consiste à appliquer notre algorithme d'optimisation sur le modèle c_q afin de détecter si ce modèle représente bien la nouvelle donnée ou si il a besoin d'être mis à jour. Nous ajustons les paramètres du modèle et mettons à jour les affectations des données. Selon les modifications obtenues après les itérations de notre algorithme d'optimisation, nous appliquons une des règles suivantes :

1. si la nouvelle donnée est associée à une composante q' non-modifiée, nous propageons notre mise à jour à ce noeud ;
2. si la nouvelle donnée est associée à une nouvelle composante, nous ajoutons cette composante à l'ensemble des fils du noeud q ;
3. si la nouvelle donnée est associée à une composante modifiée, cela implique une restructuration plus importante des données. La partie modifiée du sous-arbre de racine q est reconstruite à partir des feuilles. Les différentes étapes consistent à :
   - sélectionner les feuilles des noeuds modifiés et mettre à jour le modèle obtenu avec notre algorithme d'optimisation. Nous notons c_{new} cette partition ;
   - reconstruire un arbre binaire t, à partir du modèle c_{new} et des noeuds $q \in c_q$ non-modifiés ;
   - sélectionner les niveaux pertinents de l'arbre t avec notre algorithme 4 ;
   - mettre à jour l'arbre en remplaçant le sous-arbre de racine q par le nouveau sous-arbre t.

L'algorithme 5 page 124 explique en détail notre algorithme hiérarchique. La figure 6.3 ci-après présente un exemple de mise à jour d'un arbre avec notre méthode. Une nouvelle donnée est ajoutée par la racine de l'arbre et entraîne une modification de la structure de la partition dans un fils de la racine. Les différentes étapes de la mise à jour sont explicitées en détail.

Choix de conception

Détection des modifications dues à une nouvelle donnée

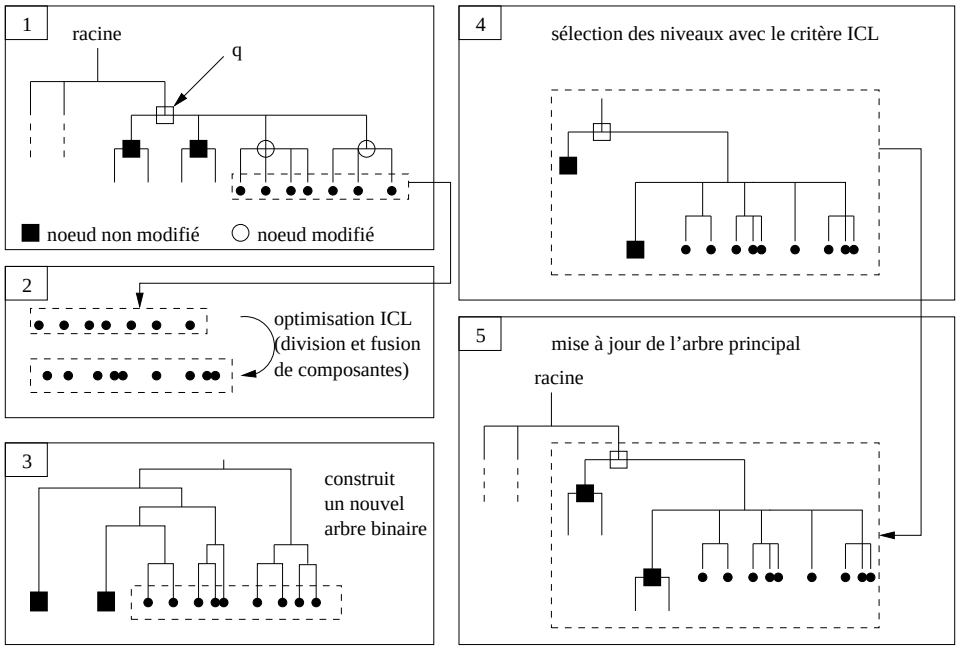


Figure 6.3: Exemple d'une mise à jour de la hiérarchie avec une nouvelle donnée : nous ajoutons une nouvelle donnée par la racine, reconstruisons le modèle c_{racine} et appliquons notre algorithme d'optimisation du critère ICL. Dans cet exemple, aucune composante n'est modifiée et la nouvelle donnée est ainsi affectée à une composante non modifiée q. La mise à jour est donc propagée à ce noeud. La figure 1 montre les noeuds modifiés et non-modifiés après l'optimisation du critère ICL. Nous supposons ici que la nouvelle donnée est associée à une composante modifiée. Nous récupérons les feuilles des noeuds modifiés et nous appliquons notre algorithme d'optimisation (figure 2). Un arbre binaire est ensuite reconstruit avec l'algorithme de Fraley (page 112) à partir des feuilles mises à jour et des noeuds non-modifiés (figure 3), et les niveaux présentant un minimum local du critère ICL sont sélectionnés. La figure 4 présente le nouveau sous-arbre obtenu. Enfin, l'arbre principal est mis à jour en remplaçant le sous-arbre de racine q par celui obtenu précédemment. Notons que les fils des noeuds non-modifiés appartenant à c_q sont conservés.

La détection des changements dus à une nouvelle donnée est réalisée à l'étape 2 de notre algorithme. Des fusions et divisions de composantes sont testées afin d'identifier les changements de structure. Nous récupérons ainsi les noeuds modifiés et non-modifiés en se basant sur les différences d'affectation des données de la partition initiale.

Dans notre proposition, nous ne cherchons que les noeuds dont les ensembles des données associées sont modifiés. D'autres types de modifications pourraient être détectés : par exemple la fusion de deux composantes ou l'affectation d'un fils d'une composante à une autre. Tous ces cas particuliers sont détectables en pratique et demandent une procédure de mise à jour particulière. Pour le cas où la nouvelle donnée est associée à une composante issue d'une fusion de deux composantes a et b initiales, nous pourrions simplement fusionner a et b dans l'arbre et propager la mise à jour à leurs fils. La détection de ces cas présente l'inconvénient de demander plus de temps de calcul. Nous avons donc décidé de limiter les recherches aux composantes seules, c'est à dire, de seulement détecter si une composante est toujours identique après sa mise à jour avec notre algorithme d'optimisation.

Mise à jour seulement des noeuds affectés par la nouvelle donnée

La mise à jour de l'arbre est simplement réalisée sur les composantes affectées par la nouvelle donnée. En effet si la nouvelle donnée est affectée à une composante non-modifiée et que plusieurs autres sont modifiées, nous ne les prenons pas en compte. De telles modifications sont généralement liées à une division de composantes, entraînant un glissement vers un niveau inférieur (problème explicité plus loin).

Une nouvelle donnée peut entraîner une reconstruction complète d'un sous-arbre, ou même de l'arbre entier dans le pire des cas. La mise à jour d'un noeud à l'étape 5 consiste à reconstruire un nouveau sous-arbre, seulement à partir des **feuilles des noeuds modifiés** mis à jour et des **noeuds non-modifiés**. Ainsi nous ne reconstruisons bien que les parties modifiées de l'arbre (les fils des noeuds non-modifiés sont conservés). Cela peut par contre sembler coûteux mais est nécessaire. En effet, une nouvelle donnée peut entièrement modifier un sous-arbre. Par exemple, un changement d'échelle dans les données peut demander une restructuration complète de certains niveaux.

Problèmes rencontrés

Nous expliquons maintenant plusieurs points pour comprendre notre algorithme. Nous avons eu à faire face à trois problèmes distincts.

Redondance des partitions

Le premier problème est lié à la mise à jour d'une partition avec notre algorithme d'optimisation afin de tester si la nouvelle donnée entraîne une modification de la partition (étape 2 de notre algorithme 5). Nous rappelons que notre arbre est composé de partitions présentant des minima locaux du critère ICL. Quand nous ajoutons une nouvelle donnée, nous optimisons ce critère à l'aide de fusions et divisions de composantes pour détecter les modifications dans une partition (les fils du noeud mis à jour). Il est possible, après une succession de divisions de composantes, que l'on obtienne une partition identique à une partition située à un niveau supérieur (nous rappelons que les niveaux vont croissant en partant de la racine). Notre algorithme d'optimisation trouverait ainsi le même optimum du critère ICL qu'un niveau plus détaillé existant et les modifications détectées entraîneraient une mise à jour inutile de notre arbre.

Afin de résoudre ce problème de redondance entre les niveaux de l'arbre, nous avons décidé de limiter le nombre de divisions testées lors de l'application de notre algorithme d'optimisation à l'étape 2.2. Nous ne permettons qu'une seule division de composantes. Cela permet d'éviter de glisser vers un niveau inférieur tout en gardant la possibilité de créer une nouvelle composante. Le nombre de fusions n'est quant à lui pas limité.

Perte de détails

Le second problème concerne la mise à jour d'une partition avec notre algorithme d'optimisation sur les feuilles de notre arbre (étape 5.2 de l'algorithme page 124). Nous avons noté que dans certains cas, la mise à jour entraînait une perte de détail au niveau des feuilles : l'algorithme d'optimisation fusionnait un nombre important de composantes en une composante comprenant beaucoup d'images. Cette perte de détail n'est pas acceptable dans notre contexte. L'analyse du problème a montré que ce comportement était dû à notre algorithme d'optimisation dans le cas où nous disposons de beaucoup de composantes associées à peu d'images. Il a été explicité en détail dans le chapitre précédent (page 89). Nous avons montré qu'un changement d'échelle pouvait provoquer une série de fusions de composantes. Si cette

propriété est intéressante pour obtenir une partition (généralisation de composantes après un changement d'échelle), elle pose un problème pour une classification hiérarchique où notre objectif est de fournir plusieurs niveaux de détail.

La solution a consisté à stabiliser le modèle avant de lancer notre procédure d'optimisation sur les feuilles du sous-arbre mis à jour (étape 5.2 de l'algorithme page 124). Nous avons décidé de rajouter une nouvelle composante dans le modèle, associée à la nouvelle donnée. Cette nouvelle composante permet d'éviter que l'optimisation du critère ICL sur le modèle initial ne provoque trop de changement. Puisque la nouvelle donnée est fortement associée à une composante, elle n'a aucun impact sur les paramètres des autres composantes.

Résumé des partitions

Enfin, le dernier problème est lié à la faculté de notre algorithme de trouver des niveaux toujours plus généraux. En effet, pour chaque nouvelle donnée, nous réalisons des mises à jours locales. Le problème est de déterminer si la nouvelle donnée peut entraîner un changement de la structure de la collection, justifiant un niveau plus général que l'existant. Par exemple si la racine est mise à jour, il serait nécessaire ensuite de vérifier si nous pouvons généraliser encore plus la partition de ses fils : pratiquement cela consiste à reconstruire un arbre binaire à partir de la partition à généraliser et à rechercher des niveaux présentant un minimum du critère ICL. Une telle recherche est néanmoins coûteuse : nous ne pouvons pas la lancer pour chaque noeud mis à jour.

Nous faisons l'hypothèse que cette recherche n'est nécessaire que dans le cas où la nouvelle donnée entraîne beaucoup de modifications dans les fils du noeud mis à jour. Cette recherche est donc effectuée seulement quand la mise à jour nécessite la reconstruction d'un sous-arbre, à l'étape 5.5 de notre algorithme. La technique est de reconstruire un arbre binaire à partir **des feuilles des noeuds modifiés**, après leur mise à jour, et **l'ensemble des noeuds non modifiés**. La sélection des niveaux pertinents à l'aide de notre algorithme 4 est réalisée sur l'ensemble des fils du noeud mis à jour.

Il est bien entendu possible que des niveaux plus généraux ne soient pas détectés si les mises à jour ne passent jamais par l'étape 5 de notre algorithme. Nous espérons que dans la pratique ce cas soit rare. Pour que cette recherche soit lancée, il suffit que la nouvelle donnée soit associée à une composante modifiée.

Complexité

Dans le pire des cas, il est nécessaire, pour chaque nouvelle donnée de reconstruire entièrement l'arbre à partir des feuilles. La complexité est ainsi en $O(K^2 + K.d^2.n.E)$, où K est le nombre de classes dans la partition obtenue aux feuilles de l'arbre et E le nombre d'itérations de l'algorithme EM. Le temps de calcul est linéaire en fonction du nombre de données. Deux points sont de plus à noter :

- en pratique une reconstruction complète de l'arbre (reconstruire l'arbre binaire à partir de toutes les feuilles) est peu fréquente ;
- le nombre d'itérations E de l'algorithme EM est limité puisque nous l'appliquons sur une partition obtenue à $t-1$: la convergence de l'algorithme est rapide.

Nous verrons dans les expériences que les mises à jour sont généralement locales (cela dépend de la structure de la collection). Notre algorithme présente donc une complexité acceptable si on le compare

aux algorithmes utilisés pour regarder des vidéos sur les terminaux mobiles. De tels algorithmes doivent décoder des données en flux tendu et respecter des contraintes en temps réel. Notre algorithme peut quant à lui tourner en tâche de fond en fonction de la liberté des ressources de l'appareil mobile. Nous n'avons pas une forte contrainte de temps pour mettre à jour la classification.

6.5 Expériences sur la sélection de niveaux avec le critère ICL

Nous présentons dans cette partie plusieurs expériences pour valider notre algorithme de sélection de niveaux (algorithme 4). Le chapitre suivant sera consacré entièrement à des expériences avec notre algorithme hiérarchique incrémental.

Nous construisons des arbres binaires à partir des partitions obtenues dans le chapitre précédent (pour les collections artificielle réaliste et réelle) et appliquons notre algorithme de sélection de niveaux.

6.5.1 Expérience sur la collection artificielle réaliste

Nous appliquons notre algorithme de sélection de niveaux sur les partitions temporelles et spatiales de la collection artificielle réaliste obtenues avec notre algorithme d'optimisation du critère ICL.

Collection temporelle

La figure 6.4 page 125 présente l'arbre obtenu à partir du modèle de mélange des données temporelles réalistes (partition représentée par la figure 5.7, page 97). L'arbre est composé de 3 niveaux et est équilibré. L'algorithme de sélection de niveaux a ainsi permis ici de trouver une seule partition plus générale que celle obtenue avec notre algorithme d'optimisation, représentée par l'ensemble des fils de la racine. Les noeuds 3, 6, 7 et 8 n'ont pas de fils, ce sont donc les composantes initiales du modèle.

Des exemples de partitions obtenues dans l'arbre sont présentées par la figure 6.5, page 125. La figure (a) présente le niveau le plus général. Nous avons réussi à trouver une solution plus générale et qui semble cohérente. Les composantes 1, 2, 4 et 5 regroupent plusieurs classes du modèle initial. Les figures (b), (c) et (d) présentent respectivement les fils des composantes 5, 4 et 2. Chacune de ces partitions peut être retrouvée dans le modèle initial puisque ce sont les feuilles de l'arbre.

Collection spatiale

L'arbre obtenu pour la classification spatiale est présenté par la figure 6.6, page 126. Il est composé de 4 niveaux et est un peu déséquilibré, mais cela correspond à la structure de la collection : le noeud 1 comprend un structure plus riche que les autres et a besoin d'être plus détaillé.

L'algorithme a permis de déterminer 2 niveaux plus généraux que la partition obtenue précédemment (figure 5.9, page 99). Le second niveau, présenté dans la figure 6.7(a) page 126, est composé de 4 classes distinctes et définit bien la structure de la collection. Le troisième niveau, obtenu pour les fils du noeud 1, est détaillé par la figure (b). La partition est composée de classes distinctes et semble correcte pour un niveau intermédiaire. Les figures (c) et (d) représentent respectivement les fils des noeuds 2 et 1.3. Ces partitions sont des sous-ensembles de la partition initiale.

6.5.2 Expérience sur la collection réelle

Nous appliquons maintenant notre algorithme sur les partitions temporelles et spatiales de la collection réelle.

Collection temporelle

La figure 6.8 page 127 présente l'arbre obtenu pour la classification temporelle. Il est composé de 4 niveaux. L'algorithme a ainsi permis de mettre en valeur un niveau intermédiaire pour la majorité des composantes initiales puisque seul un fils de la racine n'a pas de descendance (le noeud 2). Pour certaines composantes, représentées par des feuilles au quatrième niveau, nous trouvons deux niveaux intermédiaires.

La partition la plus générale, représentée par 6.9(a) page 127, est composée de 5 composantes. Les composantes sont distinctes et chaque limite est visuellement justifiée. Notons que dans la partition initiale (figure 5.13, page 102) , nous obtenions 19 classes. La figure 6.9(b) présente les fils du noeud 3. Cette partition est essentiellement composée de classes obtenues dans le modèle initial. Seules trois classes sont plus générales et contiennent des fils. Nous obtenons une partition présentant une sur-segmentation plus faible que la partition initiale. Les figures (c) et (d) représentent respectivement les fils des composantes 1 et 5 et sont des sous-ensembles de classes de la partition initiale.

Collection spatiale

La recherche de niveaux plus généraux à partir de la partition spatiale (figure 5.15, page 104) n'a donné aucun résultat. L'arbre obtenu est composé de deux niveaux : la racine et ses fils, où chaque composante obtenue précédemment est représentée par une feuille.

Ce résultat est due à la structure particulière des données. En effet l'utilisateur a pris beaucoup d'images sur des lieux précis, entraînant un nombre important d'images avec des coordonnées similaires. Nous avons ainsi obtenu des classes compactes, concentrées en un seul point dans la partition initiale. Une telle classe présente une densité élevée, toutes les données étant sur le centre de la composante. Le regroupement de deux classes proches, présentant cette configuration, n'est pas très pertinent : la densité est faible, les données n'ayant aucune ressemblance avec une distribution gaussienne. Ces données ne peuvent pas être bien définies par un modèle de mélange gaussien plus général, ce qui explique l'absence de minimum local du critère ICL dans l'arbre binaire obtenu avec l'algorithme 3. Nous proposons dans la suite une technique simple pour régler le problème.

Pré-traitement sur les métadonnées spatiales

Pour résoudre ce problème, nous proposons de nous focaliser sur les lieux de prises de vue. Notre nouvel objectif est de proposer une organisation des différents lieux présents dans la collection, et non plus une classification des images. Le nombre d'images prises dans un lieu ne rentre ainsi plus en compte dans le processus de classification. Notre solution consiste à réaliser un pré-traitement sur la collection : nous la résumons en regroupant les images avec les mêmes coordonnées géographiques. En pratique, ce pré-traitement dépend du dispositif de mesure de la localisation, par exemple la sensibilité du GPS. Il est nécessaire de fixer un seuil arbitraire pour définir ce que sont deux lieux identiques. Pour donnée un ordre de grandeur à ce seuil, le système GALILEO va apporter une précision de l'ordre du mètre.

Chaque coordonnée classée par notre algorithme peut représenter un ensemble d'images, et non plus une seule image. Pour chaque nouvelle image rajoutée dans la collection, nous vérifions tout d'abord si le

lieu de prise de vue n'est pas déjà présent dans la hiérarchie. Dans le cas où il est présent, nous rajoutons simplement l'image dans la feuille de l'arbre associée à ce lieu. Dans le cas contraire, nous lançons notre processus de mise à jour. Ce pré-traitement pour les métadonnées permet de faire disparaître les classes avec une forte concentration de données sur un lieu précis. Nous verrons que la détection de résumés dans la partition spatiale est sensiblement améliorée.

Dans ces deux expériences et hormis pour le cas des métadonnées spatiales réelles, les niveaux sélectionnés en se basant sur le critère ICL présentent de bonnes propriétés : les composantes sont distinctes et la structure des partitions sont des solutions cohérentes pour proposer différentes vues de la collection. Les arbres obtenus ont des paramètres variables (nombre de fils, profondeur, largeur), déterminés à partir de la structure des données.

6.6 Conclusion

Nous avons tout d'abord présenté plusieurs algorithmes hiérarchiques liés aux modèles de mélange. Parmi les différents choix, l'algorithme agglomératif de Fraley [44] semble le plus pertinent pour notre cas d'utilisation. Contrairement aux autres solutions, il n'est pas déterministe et ne dépend pas de paramètres critiques.

L'arbre fourni en sortie par un algorithme agglomératif étant binaire, ce type d'arbre n'est pas adéquat pour parcourir une collection d'images. Nous avons proposé une technique de sélection pour choisir les partitions de l'arbre de bonne qualité. La méthode consiste à comparer les différentes partitions de l'arbre à l'aide du critère statistique ICL et à sélectionner les niveaux présentant un minimum local.

Enfin nous avons combiné notre algorithme incrémental d'optimisation , l'algorithme agglomératif de Fraley et notre algorithme de sélection de niveaux pour fournir un algorithme hiérarchique incrémental. Notre approche présente les avantages suivant :

- la qualité des partitions de l'arbre est favorisée par notre sélection de niveaux avec le critère ICL ;
- les propriétés de l'arbre sont libres et déterminées par la structure des données. La profondeur, la largeur et le nombre de fils par noeud sont libres. L'algorithme dépend peu de paramètres arbitraires à régler manuellement ;
- la complexité de l'algorithme est linéaire en fonction du nombre de données.

Nous proposons dans le chapitre suivant d'expérimenter notre algorithme sur les collections d'images présentées au chapitre 5.

Algorithme 5 Algorithme hiérarchique incrémental

initialisation : le noeud courant est la racine de l'arbre : $q = racine$. Ajouter la nouvelle donnée new dans l'arbre.

si c_q est vide **alors**

1.ajouter new dans le noeud q et tester des divisions de la composante afin d'obtenir une classification plus fine. Un nouveau fils est ajouté à q pour chaque division de composantes obtenue. Aller à l'étape 6.

sinon

2.1.récupérer le modèle m associé à c_q ;

2.2.mise à jour du modèle m en ajoutant la nouvelle donnée new et en itérant notre algorithme d'optimisation (algorithme 2, page 88). Le nombre de divisions à tester est ici limité à 1 ;

2.3.récupérer les composantes modifiées $Q_{mod} \in c_q$ et non modifiées $Q_{n-mod} \in c_q$.

si new est associé à une composante $q' \in Q_{n-mod}$ **alors**

3.$q = q'$ et aller à l'instruction 1.

finsi

si new est associé à une nouvelle composante q' ($Q_{mod} = \emptyset$ et $Q_{non-mod} = \emptyset$) **alors**

4.ajouter le nouveau noeud $q' \in c_q$, comprenant les paramètres de q' et l'image new.

finsi

si new est associé à une composante $q' \in Q_{mod}$ **alors**

5.1.récupérer les feuilles fils des composantes modifiées. Nous notons c_{new} le modèle associé à cet ensemble de noeuds ;

5.2.mise à jour du modèle c_{new} avec notre algorithme d'optimisation. Le modèle c_{new} est initialisé en ajoutant une nouvelle composante pour la nouvelle donnée ;

5.3.construire le sous arbre t de racine q_{new} avec l'algorithme agglomératif 3 à partir du modèle c_{new} et des noeuds non-modifiés $\in Q_{n-mod}$;

5.4.optimiser l'arbre t en sélectionnant les niveaux présentant un optimum du critère ICL avec l'algorithme 4 ;

5.5.effacer les noeuds modifiés $\in Q_{mod}$ de l'arbre initial ;

5.6.mise à jour de l'arbre avec le nouveau sous arbre t optimisé. Nous remplaçons le sous-arbre de racine q par t.

finsi

finsi

6. fin de la mise à jour.

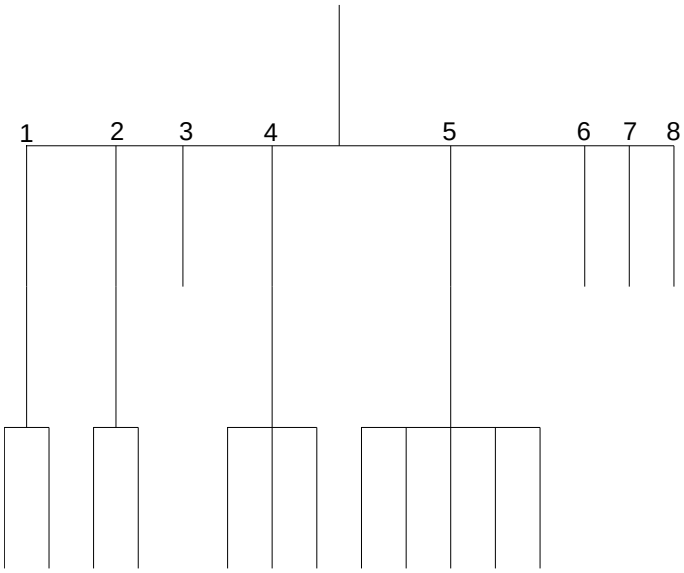


Figure 6.4: Collection artificielle réaliste : classification temporelle. L'arbre est obtenu à partir de l'algorithme de Fraley (algorithme 3) après avoir sélectionné les niveaux présentant un minimum local du critère ICL (algorithme 4). Les numéros des noeuds correspondent aux composantes de la figure 6.5. L'arbre est composé de 3 *niveaux et est équilibré. L'algorithme de sélection de niveaux a permis de trouver une seule partition plus générale que celle obtenue dans les expériences du chapitre précédent. Les noeuds* 3, 6, 7 *et* 8 *n'ont pas de fils, ce sont donc les composantes initiales du modèle.*

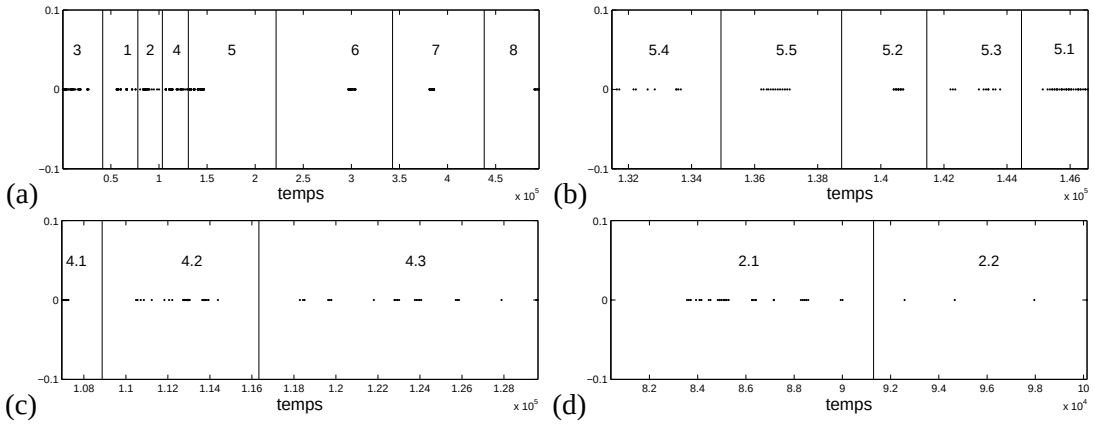


Figure 6.5: Collection artificielle réaliste : classification temporelle. Exemples de partitions sélectionnées avec l'algorithme 4. Les lignes verticales représentent les limites entre les classes. La figure (a) présente le niveau le plus général, ce sont les fils de la racine. Les figures (b), (c) et (d) sont respectivement les fils des composantes 5, 4 *et* 2 *de la figure (a).*

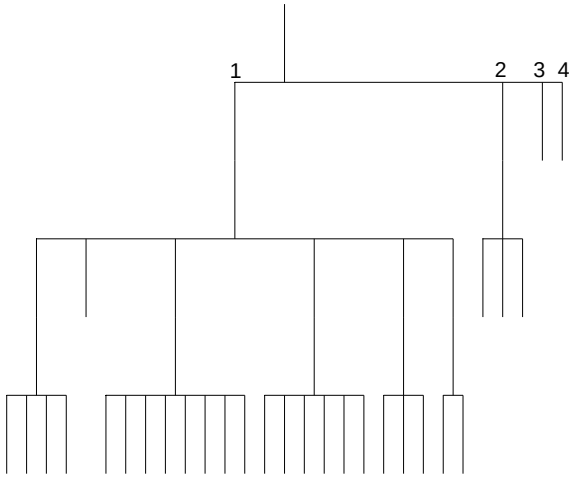


Figure 6.6: Collection artificielle réaliste : classification spatial. L'arbre est obtenu à partir de l'algorithme de Fraley (algorithme 3) après avoir sélectionné les niveaux présentant un minimum local du critère ICL (algorithme 4). Les numéros des noeuds correspondent aux composantes de la figure 6.7. L'arbre est composé de 4 niveaux et est un peu déséquilibré, mais cela correspond à la structure de la collection. Nous avons obtenu 2 niveaux plus généraux que la partition obtenue dans le chapitre précédent.

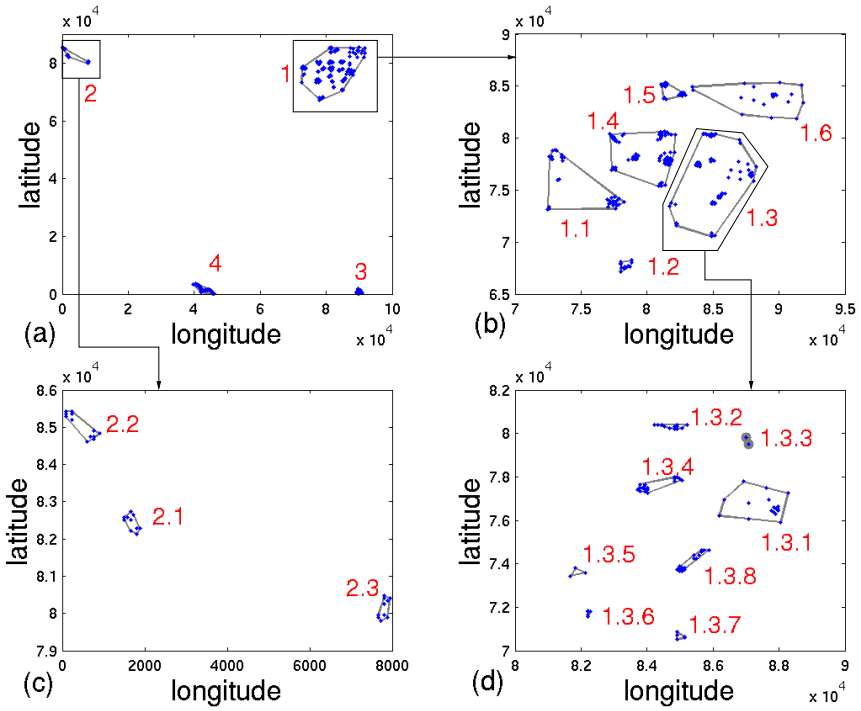


Figure 6.7: Collection artificielle réaliste : classification spatiale. Exemple de partitions sélectionnées avec l'algorithme 4. Les points représentent les métadonnées spatiales, et les "+" et les ellipses indiquent respectivement les centres et les covariances des composantes. Les polygones englobants représentent l'association des données aux composantes selon le critère MAP. Les flèches indiquent les relations parentales entre les partitions.

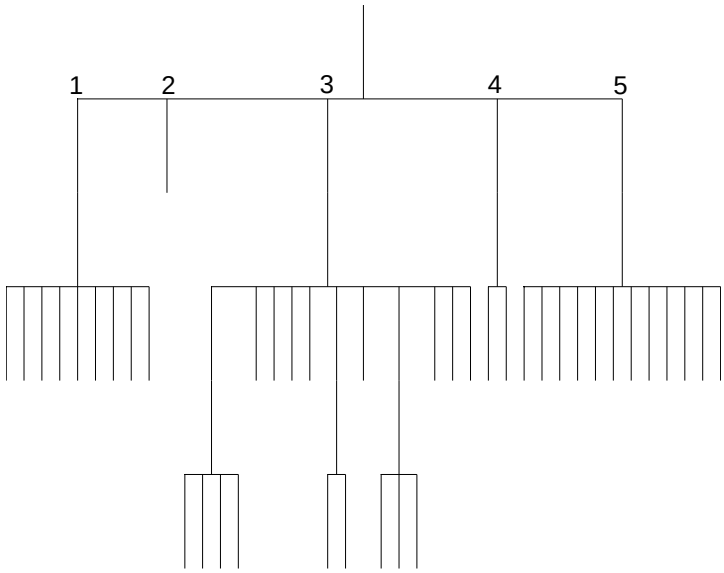


Figure 6.8: Collection réelle : classification temporelle. L'arbre est obtenu à partir de l'algorithme de Fraley (algorithme 3) après avoir sélectionné les niveaux présentant un minimum local du critère ICL (algorithme 4). Les numéros des noeuds correspondent aux composantes de la figure 6.9. L'arbre est composé de 4 *niveaux. Notre algorithme a mis en valeur un niveau intermédiaire pour la majorité des composantes initiales.*

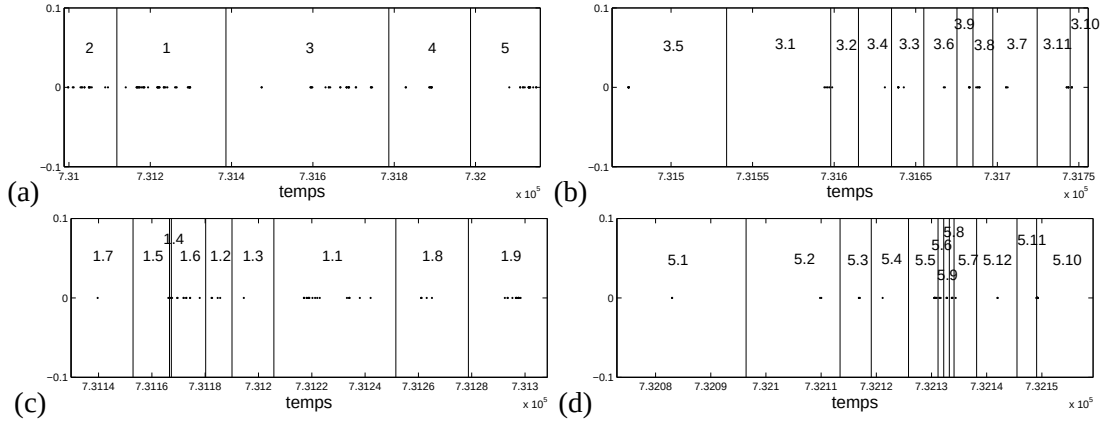


Figure 6.9: Collection réelle : classification temporelle. Exemples de partitions sélectionnées avec l'algorithme 4. Les lignes verticales représentent les limites entre les classes. La figure (a) présente le niveau le plus général, ce sont les fils de la racine. Les figures (b), (c) et (d) sont respectivement les fils des composantes 3*,* 1 *et* 5 *de la figure (a).*

CHAPITRE 7

Structuration hiérarchique incrémentale : résultats expérimentaux

Nous proposons de construire des classifications hiérarchiques à l'aide de notre algorithme, présenté dans le chapitre précédent. Les expériences sont réalisées sur la collection artificielle réaliste et la collection de Guillaume B..

7.1 Introduction

Nous présentons dans ce chapitre plusieurs expériences réalisées sur les collections d'images artificielle réaliste et réelle. Nous proposons de construire des partitions hiérarchiques à l'aide de notre algorithme présenté dans le chapitre précédent. Pour la collection réelle, nous évaluons en plus la qualité des partitions en la comparant avec une classification manuelle définie par l'utilisateur. Ce point est décisif pour évaluer notre technique puisque l'objectif premier est de fournir une partition similaire à celle réalisée manuellement.

Les premières expériences concernent l'organisation de la collection artificielle réaliste. Ensuite, les expériences sur la collection réelle sont présentées. Dans chaque expérience, nous ajoutons une à une les données (ordonnées temporellement) et les classons avec notre algorithme hiérarchique. Nous mettons en valeur les partitions obtenues et leurs différents états au cours du processus de classification.

7.2 Classifications hiérarchiques de la collection artificielle réaliste

Nous présentons dans cette section les classifications hiérarchiques obtenues avec notre algorithme pour la collection artificielle réaliste. Nous détaillons la topologie des arbres obtenus, donnons des exemples de partitions et évaluons la complexité de l'algorithme.

7.2.1 Classification temporelle

Les données temporelles sont présentées par la figure 5.5(a), page 93. La collection contient 451 images.

Topologie de l'arbre

La classification temporelle obtenue est présentée par la figure 7.2, page 136. La hiérarchie est équilibrée et très détaillée, l'arbre étant large et profond. La figure 7.1 présente différents états de la classification au cours du processus de classification (pour 90 (a), 180 (b), 270 (c) et 360 (d) données). Nous notons que l'arbre croît bien en largeur et profondeur au fur et à mesure de l'ajout des nouvelles données. Ce point est important afin de garantir un bon niveau de détail dans la classification.

Nous retrouvons des branches identiques entre les arbres (b), (c) et (d), représentées par les carrés en pointillés sur la figure 7.1. Ces arbres successifs semblent stables dans le temps. L'arbre final présente par contre peu de similarité avec l'arbre (d). Cela s'explique par la mauvaise structure du premier groupe de données (données associées aux composantes 1, 2, 3, 4, 5, 6 et 7 de la figure 7.3(a)) et par le changement d'échelle obtenu avec les groupes de données suivants. La mauvaise structure des premières données a entraîné un manque de stabilité des partitions : le changement d'échelle a modifié une grande partie des composantes, ce qui a entraîné une reconstruction importante de l'arbre. Nous avons néanmoins vérifié que les arbres sont stables dans le temps et que la majorité des données entraîne peu de modifications.

Partitions obtenues

Des exemples de partitions obtenues dans l'arbre sont présentés par la figure 7.3, page 137. Le premier niveau de l'arbre est détaillé par la figure (a). Les composantes 7, 8, 9 et 10 sont convenablement délimitées alors que les classes 1, 2, 3, 4, 5 et 6 sont discutables. Pour ces dernières le résultat est dû aux données mal structurées auxquelles elles sont associées. Néanmoins la partition semble assez pertinente pour fournir un résumé de la collection. Le nombre de classes reste limité et celles-ci contiennent chacune un nombre d'images similaires.

Les fils des composantes 5 et 10 sont représentés respectivement par les figures (b) et (c). Les classifications sont toutes les deux composées de deux classes et semblent peu pertinentes. Les fils de la composante 5.1 (figure (d)) présentent une structure similaire. Cependant ce type de partition peut être utile comme partition intermédiaire entre les parents et les fils, en permettant d'éviter un nombre excessif de fils.

Une reconstruction totale de l'arbre à partir des feuilles avec l'algorithme de Fraley (algorithme 3, page 112) et en sélectionnant les niveaux avec le critère ICL a montré que ce type de partitions n'apparaissaient plus dans la classification. Nous pouvons donc en déduire que ces partitions sont essentiellement dues à notre méthode de mise à jour incrémentale. Dans cette expérience, certaines parties de l'arbre ne sont jamais remises en cause durant tout le processus de classification. Cela est bénéfique du point de vue de la complexité en temps de calcul mais présente un désavantage sur les partitions obtenues. Néanmoins, il est possible que l'arbre soit remis entièrement à jour si une nouvelle donnée entraîne beaucoup de modifications sur les fils de la racine. Notre méthode de mise à jour permet de remettre en question les classifications seulement dans le cas où de grandes modifications sont détectées.

Les fils des composantes 10.1 et $5.1.2$ sont présentés respectivement par les figures 7.3(e) et (f). Contrairement à leur père, les partitions sont ici de bonne qualité. Les limites entre les classes sont visuellement justifiées.

Complexité

La complexité de notre algorithme peut être évaluée comme suit. Pendant la construction de notre classification temporelle, l'algorithme agglomératif permettant de régénérer un arbre binaire a été appelé dans 57% des cas, et pour chaque cas concernait en moyenne 33% des données. L'ajout d'une nouvelle donnée entraîne dans la majorité des cas des modifications, mais seulement sur une petite partie des données.

7.2.2 Classification spatiale

Les données spatiales sont présentées par la figure 5.5(b), page 93. Le pré-traitement sur les données spatiales a fourni 410 lieux distincts. Nous avons éliminé 41 données, présentes en plusieurs exemplaires dans la collection.

Topologie de l'arbre

La figure 7.5 présente la topologie de l'arbre obtenu, page 138. Il est bien équilibré et le nombre de fils par noeud varie entre 2 et 10. La figure 7.4 présente les différents états de la classification au cours du processus de classification. Nous observons une bonne stabilité dans les arbres successifs : les arbres (a), (b), (d) et l'arbre final présentent des branches similaires. Nous notons dans les premiers temps que l'arbre croît en largeur et en profondeur au fur et à mesure de l'ajout des nouvelles données (figures 7.4(a), (b) et (c)). Il devient ensuite moins dense sur la fin des données, comme nous pouvons le constater sur (d) et l'arbre final (figure 7.5). Cette simplification est due au changement d'échelle dans les données, entraînant généralement des re-structurations importantes de l'arbre. L'arbre (c) ne comprend que des données incluses dans la classe 5 de la figure 7.6(a) page 139 et présente beaucoup de détails sur leurs structures. Le parcours de cet arbre a même montré que nous obtenions trop de détails. Une fois que les autres données sont rajoutées, le changement d'échelle entraîne des fusions de composantes, ce qui simplifie la hiérarchie.

Partitions obtenues

Le niveau le plus général de la classification finale est présenté par la figure 7.6(a), page 139. La partition est composée de 6 classes et est similaire à la partition obtenue dans le chapitre précédent (figure 6.7(a), page 126). La seule différence vient des classes 1, 2 et 6 qui ne sont pas regroupées ensembles. Un tel regroupement aurait été préférable. Nous avons vérifié si un niveau plus général regroupant ces trois composantes existait (cela revient à reconstruire un arbre binaire à partir des fils de la racine, et de rechercher les minima locaux du critère ICL dans ce sous-arbre) afin de vérifier si l'erreur venait de notre algorithme. Ce test a montré qu'il n'existait pas de niveau plus général à partir des fils de la racine. Par contre, en reconstruisant l'arbre à partir de ses feuilles, nous obtenons un arbre avec un niveau comprenant seulement 4 composantes pour les fils de la racine (les classes 1, 2 et 6 sont regroupées en une composante). La mise à jour incrémentale présente ainsi un léger désavantage par rapport à une reconstruction totale de l'arbre. Ce défaut est néanmoins normal puisque nous réalisons seulement des mises à jours locales.

Les fils de la composante 5 sont détaillés par la figure 7.6(b). Nous obtenons une partition de 6 composantes. Les classes sont distinctes et certaines sont assez larges. La partition présente ainsi un niveau intermédiaire pertinent : le niveau de détail reste limité et permet bien de choisir entre des ensembles de classifications plus précis. Les fils de la composante 5.1, 5.2 et 5.3 sont présentés respectivement par les

figures 7.6(e), (c) et (f). Chacune de ces partitions présente des composantes distinctes avec des données bien regroupées. Nous notons une similarité avec la partition obtenue avec notre algorithme incrémental (figure 6.7(b), page 126).

Enfin les fils de la composante 4 de la partition (a) sont représentés par la figure 7.6(d). La partition met bien en valeur la structure des données. Nous notons que notre algorithme d'optimisation n'a pas fourni un tel niveau de détail sur ce sous-ensemble de données (figure 5.9(d), page 99, la composante de droite regroupe les fils de la composante 4).

Complexité

Du point de vue performance, l'algorithme agglomératif a été appelé dans 30% de cas, et pour chaque cas concernait en moyenne 29% des données. Comme pour le cas temporel, la mise à jour concerne en général un petit sous-ensemble des données.

7.3 Classifications hiérarchiques de la collection réelle de Guillaume B.

Nous proposons des classifications hiérarchiques spatiale et temporelle obtenues avec notre technique, à partir de la collection réelle de Guillaume B.. Ensuite, nous proposons une évaluation pratique de la collection.

7.3.1 Classification temporelle

Les données temporelles sont présentées par la figure 5.6 (a), page 94. La collection contient 721 images.

Topologie de l'arbre

L'arbre final est représenté par la figure 7.8, page 140. Il est composé de 4 niveaux et est bien équilibré. Le nombre de fils par noeud varie entre 2 et 14. Contrairement à l'arbre temporel de l'expérience précédente, l'arbre comprend moins de détails alors que le nombre d'images est supérieur. Les données de la collection réelle présentent une distribution plus stable. La figure 7.7 montre les différents arbres obtenus après l'ajout de 150 (a), 300 (b), 450 (c) et 600 (d) images. La classification obtenue croît en largeur et en profondeur au fur et à mesure que les nouvelles images sont ajoutées. La stabilité de l'arbre au cours du processus de classification est correcte puisque la majorité des branches est similaire au cours du processus de classification (carrés en pointillés). Seule une minorité d'images implique une sérieuse re-structuration de l'arbre. Ce point montre bien que les données réelles présentent une structure plus stable que les données artificielles réalistes.

Partitions obtenues

La figure 7.9 page 141 présente des partitions obtenues à différents niveaux de notre arbre. La figure 7.9(a) montre le niveau le plus général. Cette partition contient 5 composantes. Les composantes 3, 4 et 5 semblent bien délimitées contrairement aux composantes 1 et 2 où les limites sont un peu plus floues. Ce niveau présente néanmoins une partition pertinente pour résumer la collection : le nombre de composantes est correct et les classes bien équilibrées du point de vue du nombre d'images. Les

composantes 2, 3, 4 et 5 de la figure 7.9(a) sont respectivement détaillées par les figures 7.9 (c), (e), (d) et (b). Les fils des composantes 4 et 3 fournissent des partitions bien définies puisque les événements temporels sont mis en valeur. Pour les composantes 2 et 5, les épisodes sont correctement trouvés mais nous pouvons noter une sur-segmentation, par exemple les groupes 2.9 et 2.10 ou 5.2, 5.4 et 5.3, due certainement à une trop grande vraisemblance des petits échantillons associés à une composante. Enfin les fils de la composante 4.4 de la figure 7.9(d) sont détaillés par la figure 7.9(f). La partition obtenue semble correcte.

Complexité

Pendant la construction de notre classification temporelle, l'algorithme agglomératif permettant de régénérer un arbre binaire a été appelé dans 23% de cas, et pour chaque cas concernait en moyenne 29% des données. Ces performances sont bonnes et montrent encore une fois que notre processus de mise à jour est pertinent. Il faut noter que ces performances sont fortement liées à la structure des données. Pour l'expérience temporelle précédente, où les données sont moins stables, les performances sont moins satisfaisantes.

7.3.2 Classification spatiale

Les données spatiales sont présentées par la figure 5.6(b) , page 94. Le pré-traitement sur les données spatiales (voir section 6.5.2, page 122) a fourni seulement 135 lieux distincts alors que nous disposions initialement de 721 données. L'utilisateur a pris des images sur seulement 135 lieux distincts. En plus d'améliorer les résultats comme nous le verrons dans la suite, ce pré-traitement présente l'avantage de limiter sensiblement le coût de calcul : le nombre de données à classer est très diminué. Nous ne présentons d'ailleurs pas l'évolution de la topologie de l'arbre, le nombre de données étant trop peu élevé.

Topologie de l'arbre

Les figures 7.10 et 7.11 page 142 montrent respectivement la classification hiérarchique spatiale et des exemples de partitions obtenues à différents niveaux. L'arbre obtenu est composé de 3 niveaux et le nombre de fils par noeud reste modéré, variant de 2 à 6. La classification présente beaucoup de détail : nous obtenons 50 feuilles pour 135 données. Contrairement à la hiérarchie obtenue dans le chapitre précédent à partir de ces mêmes données non résumées, nous obtenons ici plusieurs niveaux et peu de feuilles directement sous la racine de l'arbre.

Partitions obtenues

Le niveau le plus général est présenté dans la figure 7.11(a), et la figure 7.11(b) est un zoom sur les composantes 1, 2 et 3. La partition contient 12 composantes distinctes et compactes, ce qui est conforme aux caractéristiques des données puisque les lieux sont pour la plupart isolés. Le nombre de classes dans ce niveau reste limité et fournit un bon résumé de la collection.

Nous notons que notre algorithme d'optimisation a tendance à regrouper les données isolées, comme on peut le voir avec la composante 3 sur la figure 7.11(b). Cet aspect peut être néanmoins pertinent pour ensuite parcourir la collection (regroupement des lieux isolés adjacents).

Les fils des composantes 2 et 7 sont respectivement présentés dans les figures 7.11(d) et (c). Les deux classifications sont pertinentes puisque les groupes semblent visuellement justifiés.

Les partitions obtenues sont plus générales que celles obtenues à partir de la partition obtenue avec

Rappel	98%
Précision	86%

Table 7.1: Évaluation pratique de la collection réelle : scores de rappel *et de* précision *obtenus.*

notre algorithme d'optimisation. L'absence des nombreuses données fortement concentrées en un seul point a permis de regrouper plusieurs lieux proches en une seule classe. Il faut bien noter ici qu'un point sur les figures peut représenter plusieurs images. Ainsi la composante 3 sur la figure 7.11(b) a en réalité trois fils puisqu'il contient trois lieux distincts comprenant chacun plusieurs images.

Complexité

L'algorithme agglomératif a été appelé dans 59% des cas, et concernait pour chaque cas en moyenne 29% des données. Les appels élevés de l'algorithme agglomératif sont dus au manque de stabilité dans les partitions pour les premières données du flux. L'échelle des données varie beaucoup et cela entraîne des re-structurations importantes.

7.3.3 Évaluation pratique des partitions par l'utilisateur

Nous proposons dans cette partie une évaluation pratique de l'organisation de la collection réelle en la comparant avec une classification fournie par l'utilisateur. Dans le cas de la partition temporelle, nous utilisons une heuristique usuelle en recherche d'informations, la précision et le rappel :

$$\textit{précision} = \frac{\text{limites détectées correctement}}{\text{total des limites détectées}} \tag{7.1}$$

$$rappel = \frac{\text{limites détectées correctement}}{\text{total des limites réelles}} \tag{7.2}$$

Évaluation de la partition temporelle

Guillaume B. a classé manuellement ses images en 58 épisodes temporels. Nous avons noté qu'il regroupait généralement des images de vacances ou des images isolées successives dans un même groupe (même si le groupe obtenu se déroule sur plusieurs semaines). Le tableau 7.1 résume les scores obtenus pour la *précision* et le *rappel*.

Pour comparer avec notre résultat, nous avons récupéré toutes les feuilles de notre classification temporelle pour obtenir la classification la plus détaillée. Elle est composée de 107 classes et nous avons obtenu 57 limites correctes. Nous avons ainsi réussi à retrouver les différents événements de la collection. Puisque nous fournissons des classifications hiérarchiques, nous ne calculons pas la *précision* à partir de la partition des feuilles. Nous proposons de vérifier dans l'arbre temporel si les feuilles sont correctement regroupées dans un sous arbre approprié (si tous les groupes obtenus manuellement sont représentés par un noeud de l'arbre). Nous avons retrouvé 50 épisodes correctement définis dans notre arbre. Les erreurs sont dues aux vacances ou aux images isolées successives classées dans des feuilles séparées.

Il faut noter que certaines feuilles de l'arbre temporel présentent peu d'intérêt d'un point de vue pratique. Le problème ne se pose pas pour le parcours de l'arbre puisque l'utilisateur peut ignorer ces classes en arrêtant son parcours à une classe supérieure et de meilleure qualité. Par contre, ces classes

inutiles peuvent nuire à la performance de notre algorithme si elles sont en trop grand nombre. Une perspective de travail serait de pouvoir évaluer la qualité des partitions. Du point de vue statistique ce problème n'est pas simple : un critère de classifiabilité tel que l'entropie ne permet pas d'évaluer si le contenu d'une classe est pertinent pour un utilisateur.

Évaluation de la partition spatiale

L'utilisateur a divisé sa collection en 25 lieux différents. Le nombre élevé de feuilles dans notre arbre spatial est dû essentiellement à des images prises lors de balades, qui entraînent de nombreuses classes. Nous avons néanmoins retrouvé 23 lieux correctement regroupés dans des noeuds distincts de notre arbre spatial. Nous avons ainsi retrouvé les différents lieux de la collection. Deux erreurs subsistent : une feuille regroupe deux lieux proches et un lieu est divisé dans deux feuilles différentes. Cette dernière erreur est due à des images prises lors d'une balade, où les données présentent peu de structure.

Du point de vue cohérence, la hiérarchie spatiale obtenue est correcte, les noeuds étant facilement interprétables. Par exemple la composante 2 (figure 7.11(b), page 142) représente les différents lieux pertinents dans la ville de l'utilisateur tandis que la composante 1 regroupe les lieux aux alentours. Mais l'utilisateur peut néanmoins être déçu par le résultat : il peut s'attendre à obtenir une partition organisée d'abord par pays, puis par ville, …. Nous pensons que notre technique peut ensuite être combinée avec un *Système d'Information Géographique* (SIG) pour fournir une partition adéquate.

7.4 Conclusion

Les différentes expériences proposées ont montré que notre algorithme est efficace pour organiser des documents personnels à partir de leur date et de leur géo-localisation. La mise à jour incrémentale prend bien en compte les changements de structures des données en ré-organisant les partitions. Celles-ci présentent de bonnes propriétés (classes distinctes) et sont dans la majorité des cas pertinentes.

Néanmoins le parcours de ces classifications nécessitent de disposer d'une IHM adéquate pour proposer un système efficace. En effet le parcours d'un arbre est généralement complexe, et dans notre cas l'utilisateur doit pouvoir en parcourir deux simultanément. En ce sens, nous proposons dans le chapitre suivant deux techniques pour combiner des partitions temporelles et spatiales.

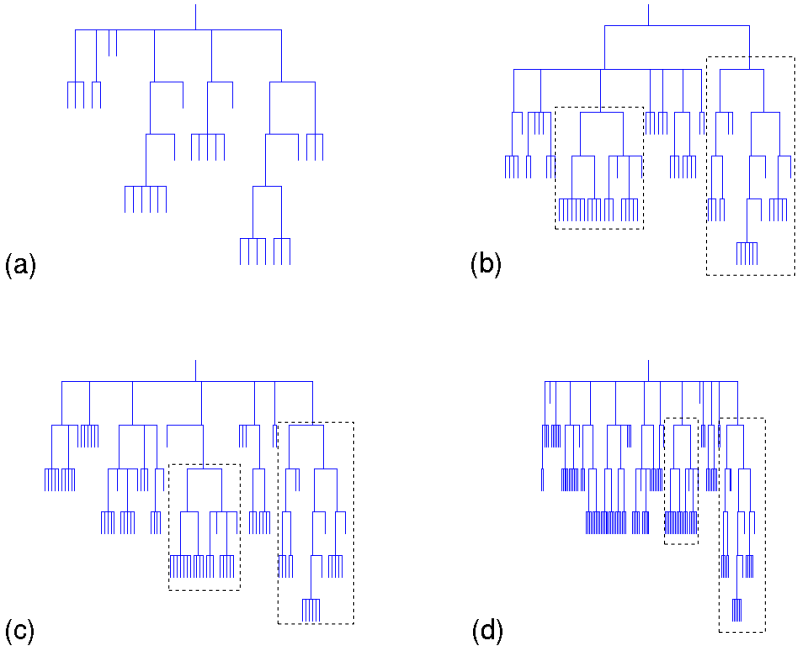


Figure 7.1: Collection artificielle réaliste : topologie de la classification temporelle obtenue après 90 *(a),* 180 *(b),* 270 *(c) et* 360 *(d) données. Les rectangles en pointillés indiquent les sections similaires des arbres deux à deux.*

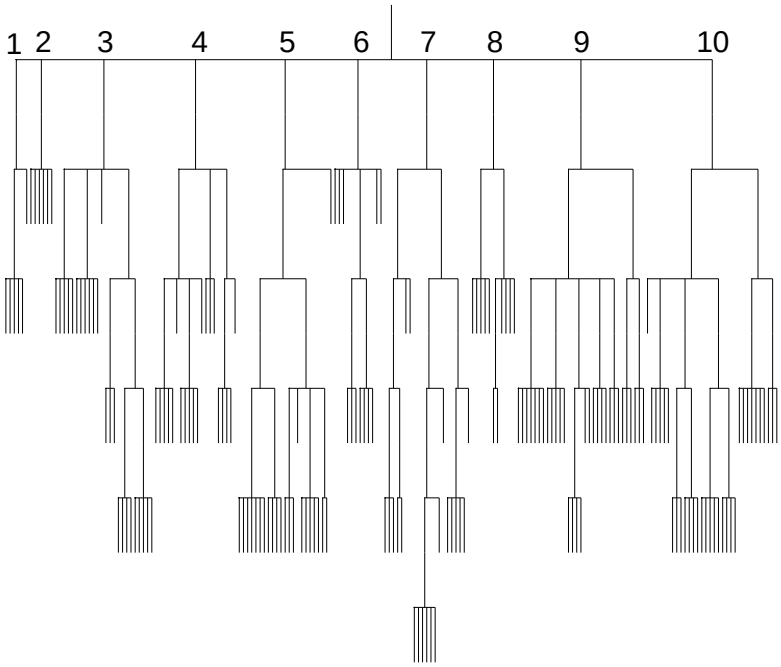


Figure 7.2: Collection artificielle réaliste : topologie de la classification temporelle obtenue avec notre algorithme hiérarchique. Les numéros sont arbitraires et font référence aux classes de la figure 7.3.

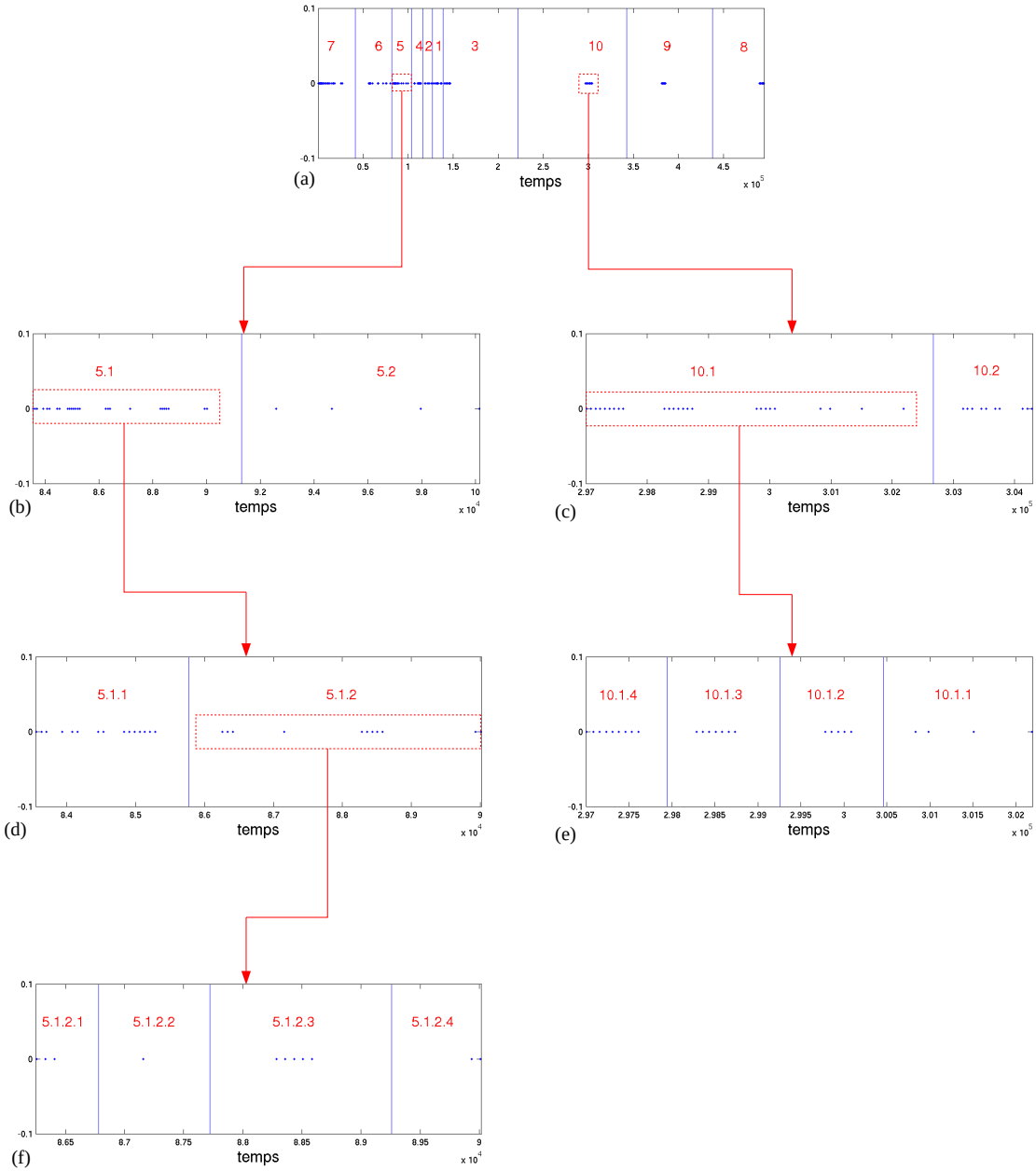


Figure 7.3: Collection artificielle réaliste : classification temporelle. Les figures présentent des exemples de partitions obtenues dans l'arbre. Chaque numéro de classe fait référence à un noeud de l'arbre de la figure 7.2. Les lignes solides sont les limites entres les classes et les points sont les données temporelles. Les flèches indiquent les relations parentales entre les partitions.

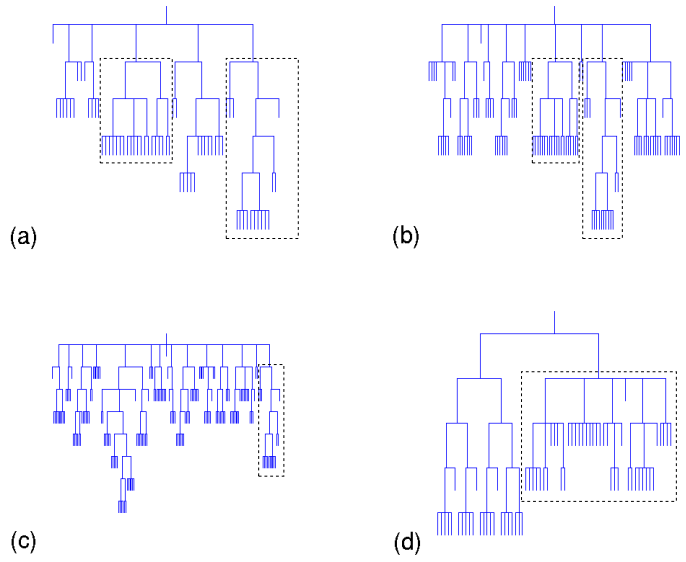


Figure 7.4: Collection artificielle réaliste : topologie de la classification spatiale obtenue après 82 (a), 164 (b), 246 (c) et 328 (d) données. Les rectangles en pointillés indiquent les sections similaires des arbres deux à deux.

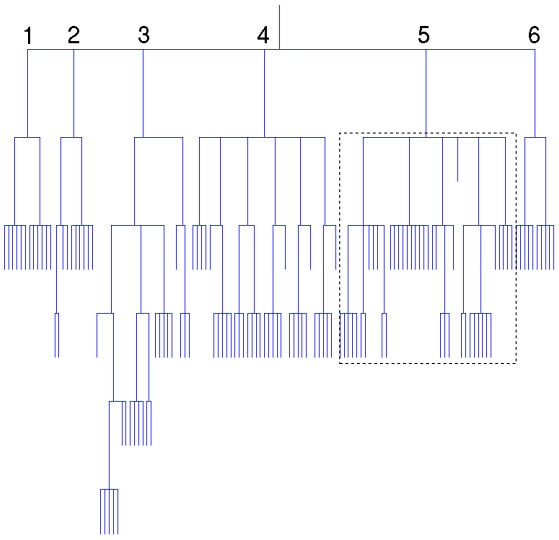


Figure 7.5: Collection artificielle réaliste : topologie de la classification spatiale obtenue avec notre algorithme hiérarchique pour la classification spatiale. Les numéros sont arbitraires et font référence aux classes de la figure 7.6. Les rectangles en pointillés indiquent les sections similaires des arbres deux à deux.

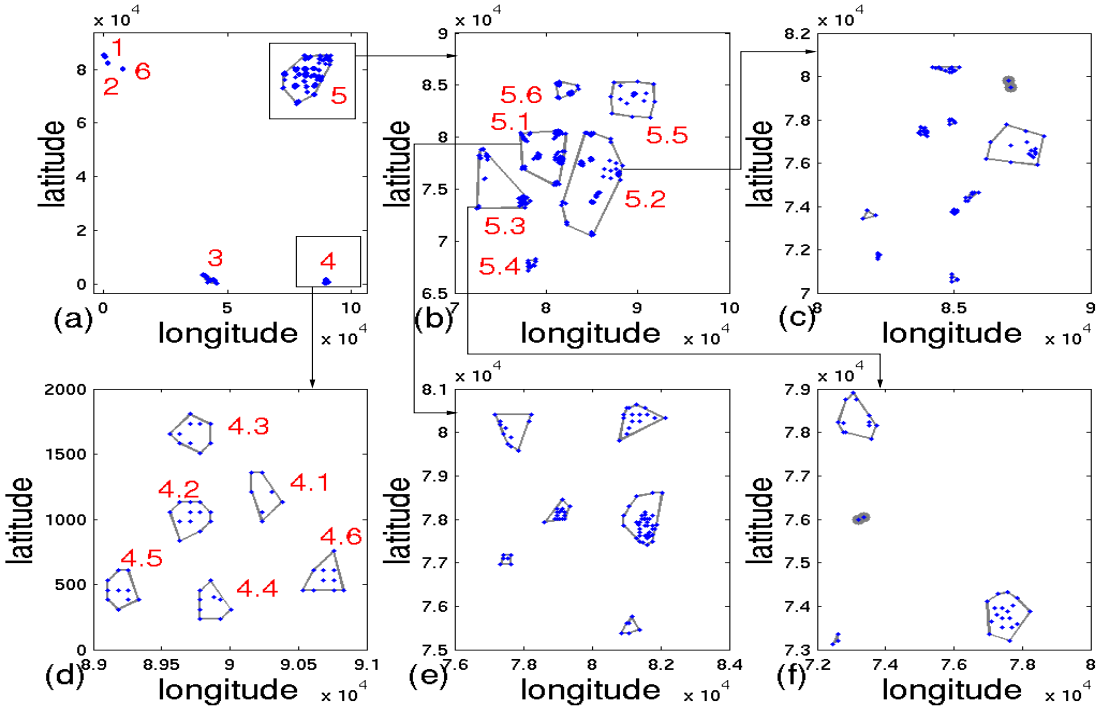


Figure 7.6: Collection artificielle réaliste : classification spatiale. Les figures représentent des exemples de partitions obtenues dans l'arbre spatial. Chaque numéro fait référence à un noeud dans l'arbre de la figure 7.5. Les points sont les données spatiales. Les "+" et les ellipses sont respectivement les centres et les variances des composantes. Les polygones englobants représentent l'association des données aux composantes selon le critère MAP. Les flèches indiquent les relations parentales entre les partitions.

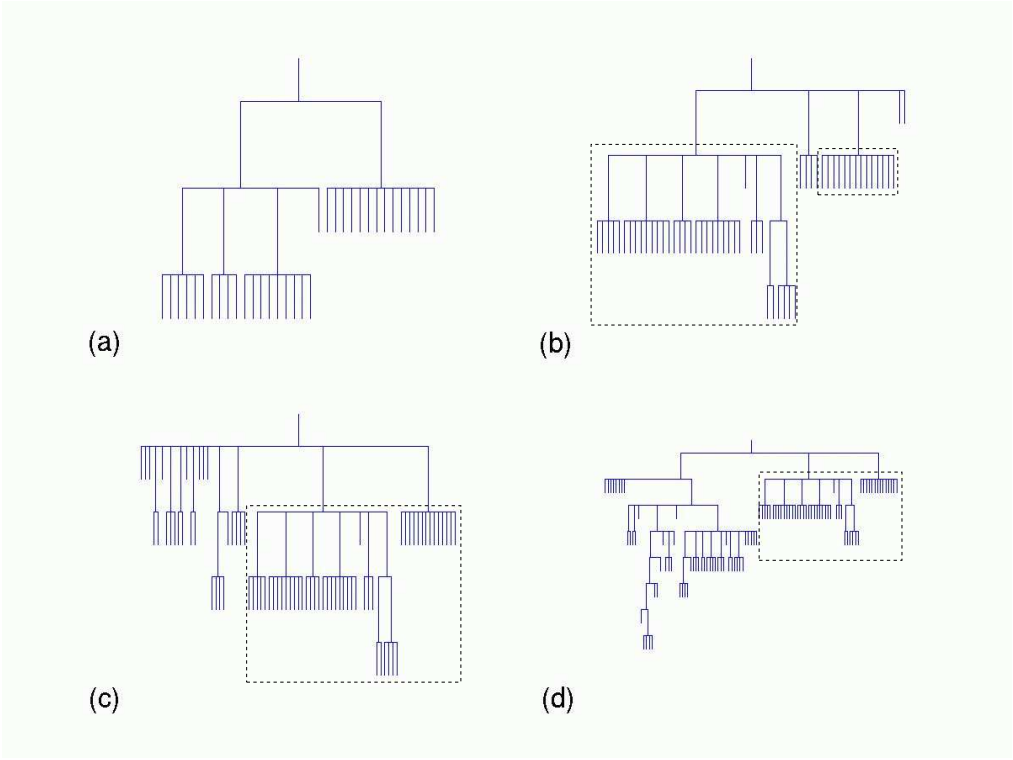


Figure 7.7: Collection réelle : topologie de la classification hiérarchique temporelle obtenue après 150 *(a),* 300 *(b),* 450 *(c) et* 600 *(d) données. Les rectangles en pointillés indiquent les sections similaires des arbres deux à deux.*

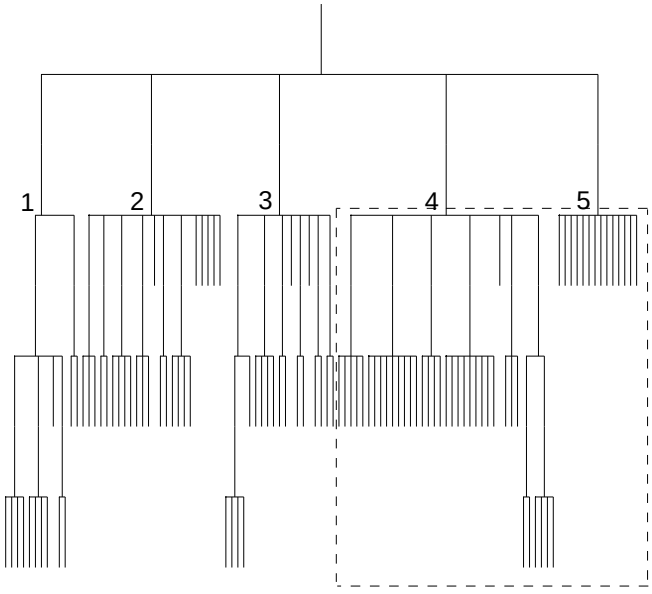


Figure 7.8: Collection réelle : topologie de la classification temporelle obtenue avec notre algorithme hiérarchique pour la classification temporelle. Chaque noeud est numéroté arbitrairement et correspond à une classe de la figure 7.9. Les rectangles en pointillés correspondent aux sections similaires avec l'arbre de la figure 7.7(d).

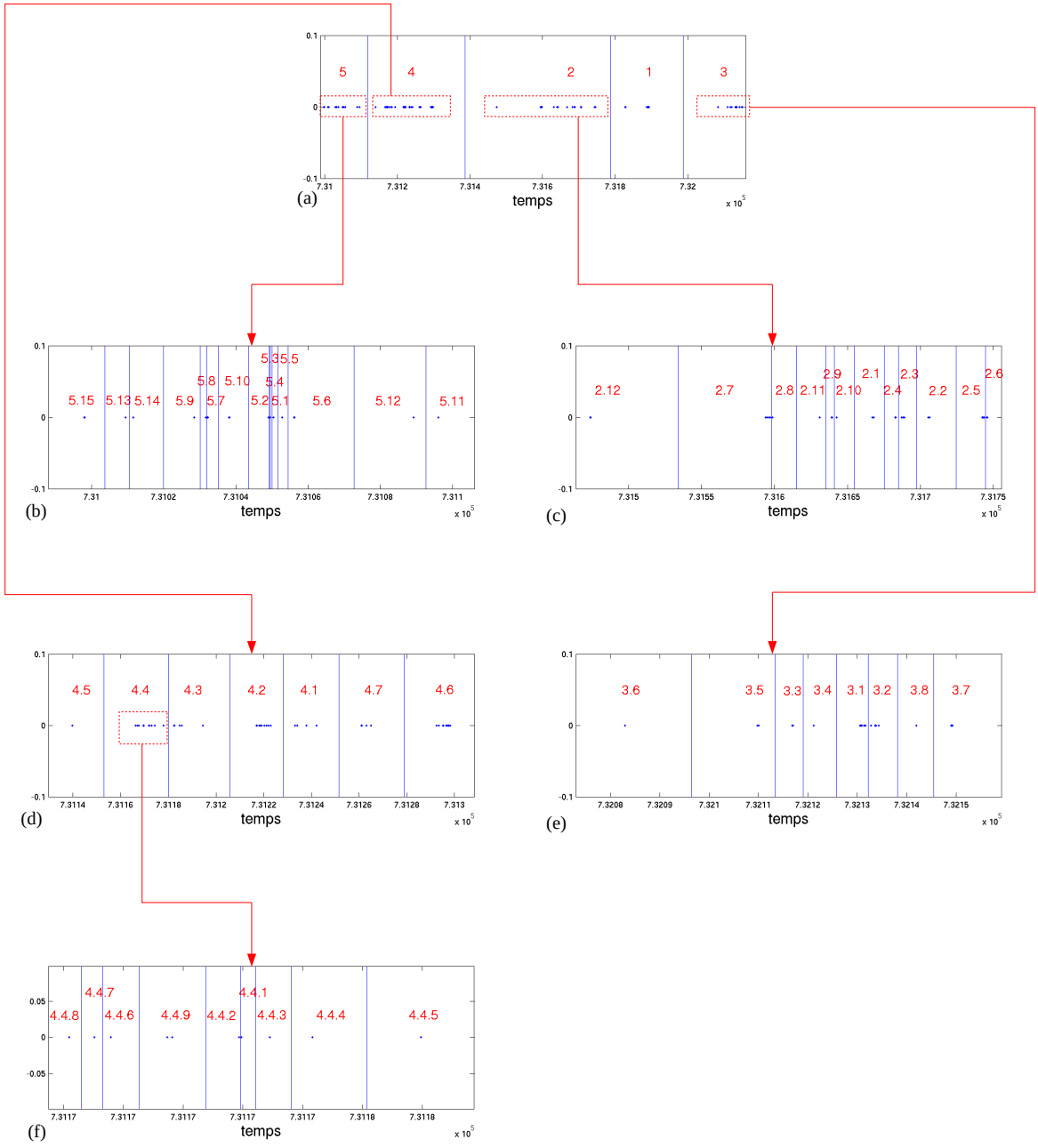


Figure 7.9: Collection réelle : classification temporelle. Les figures présentent des exemples de partitions obtenues dans la classification hiérarchique temporelle. Chaque numéro fait référence à un noeud de l'arbre présenté dans la figure 7.8. Les lignes verticales représentent les limites entre les composantes et les points indiquent les données temporelles. Les flèches indiquent les relations parentales entre les partitions.

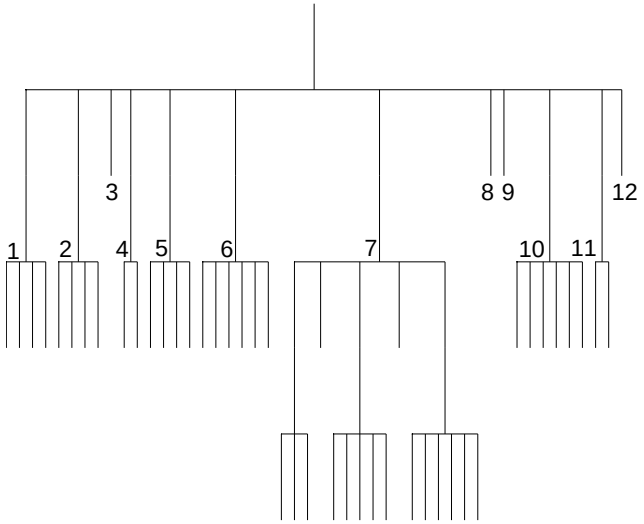


Figure 7.10: Collection réelle : topologie de la classification spatiale obtenue avec notre algorithme hiérarchique. Les numéros des noeuds correspondent aux composantes de la figure 7.11.

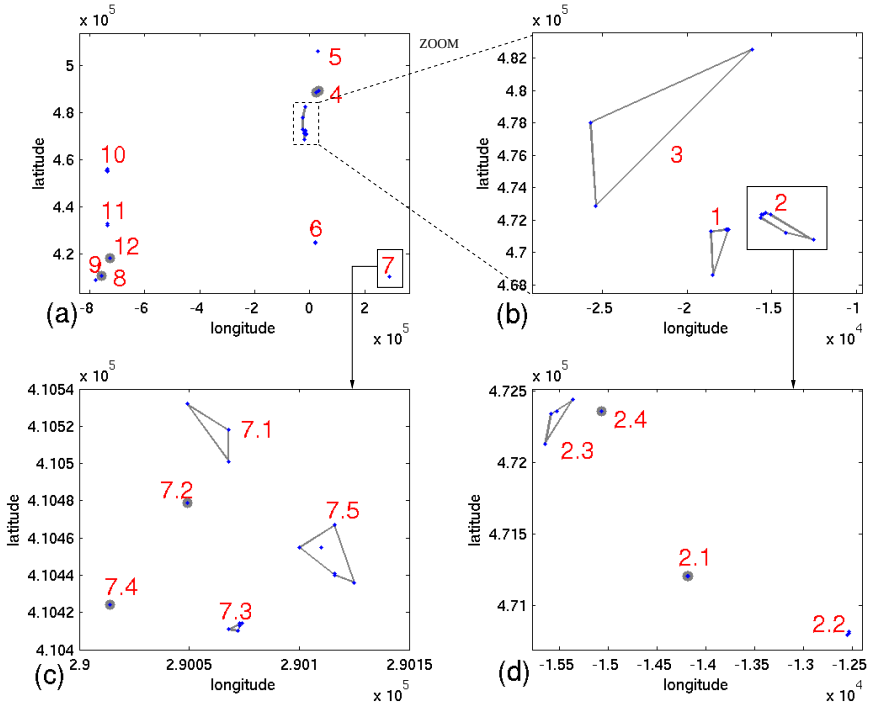


Figure 7.11: Collection réelle : classification spatiale. Les figures représentent des exemples de partitions obtenues dans la classification spatiale hiérarchique. Chaque numéro fait référence à un noeud de l'arbre présenté dans la figure 7.10. Les "+" et les ellipses sont respectivement les centres et les variances des composantes. Les polygones englobants représentent l'association des données aux composantes selon le critère MAP. Les flèches indiquent les relations parentales entre les partitions.

CHAPITRE 8

Métadonnées contextuelles & structuration jointe spatio-temporelle

Nous présentons tout d'abord dans ce chapitre une technique pour représenter les classes de nos partitions à l'aide de labels textuels, obtenus à partir des données temporelles et spatiales initiales. Ensuite, nous proposons deux procédures pour fournir des partitions hybrides afin de faciliter le parcours de la collection sur un appareil mobile.

8.1 Introduction

Les classifications temporelles et spatiales (hiérarchiques ou non) étant obtenues, il est maintenant nécessaire de fournir des partitions adaptées aux contraintes d'Interaction Homme-Machine (IHM) liées à un appareil mobile. Nous avons vu que ces appareils ont des écrans et des claviers de petite taille (limitant l'affichage simultané d'images et les interactions avec l'utilisateur) et une autonomie en énergie faible (l'affichage d'images est coûteux en énergie).

Notre contribution dans ce chapitre concerne :

- la description des classes d'images à partir d'informations contextuelles plus élaborées que les données temporelles ou spatiales initiales, obtenues à l'aide d'une base de connaissances (pour les métadonnées contextuelles temporelles) et d'un *Système d'Information Géographique* (SIG) (pour les métadonnées contextuelles spatiales). Une technique est proposée pour définir les composantes à l'aide de leurs données. La finalité est de pouvoir parcourir les classes à l'aide de résumés textuels. Cette représentation est appropriée sur un terminal mobile puisque l'affichage de texte prend peu de place sur un écran et est peu coûteux en énergie ;

- la construction d'une partition géo-temporelle obtenue à partir de partitions temporelle et spatiale **non-hiérarchiques** (obtenues avec notre algorithme d'optimisation du critère ICL, voir chapitre 5). L'objectif est de sélectionner, sur un intervalle de temps donné, les classes temporelles ou spatiales présentant la meilleure qualité. Le critère d'entropie E est choisi pour évaluer les partitions. La combinaison de ces classes fournit une partition unique. Seules les métadonnées initiales

des images sont utilisées pendant le processus de construction de cette partition hybride (pas de métadonnées contextuelles) ;

- la construction d'une partition géo-temporelle hiérarchique obtenue à partir des partitions temporelle et spatiale **hiérarchiques** (obtenues avec notre algorithme hiérarchique, voir chapitre 6). L'approche consiste à modifier la partition temporelle en la combinant localement avec la partition spatiale. Les classes temporelles sont divisées selon les classes spatiales qu'elles contiennent : les lieux sont représentés à partir des métadonnées contextuelles. La partition obtenue permet de rechercher simultanément les images par événements et lieux, à l'aide d'une IHM simple.

8.2 Représentation des classes des partitions à l'aide de métadonnées contextuelles issues d'un SIG et d'une base de connaissances

Résumer une classe à l'aide de métadonnées contextuelles est pertinent pour notre application, du point de vue de l'autonomie limitée des appareils mobiles et des contraintes d'IHM. Notre approche consiste à obtenir des informations plus riches sur les images à partir d'un SIG et d'une base de connaissances. La définition d'une classe prendra en compte les probabilités d'affectation *a posteriori* de ses images. Cette probabilité est obtenue à partir de la matrice d'affectation t des données aux composantes, définie dans la section 3.4 du chapitre 3, page 42. Une image i étant affectée à une classe selon le critère maximum *a posteriori* (MAP), elle appartient à la classe k qui a la plus forte probabilité de la générer : chaque image est associée à sa classe avec une probabilité t_{ik} que nous proposons d'utiliser pour définir nos classes à l'aide de métadonnées contextuelles. L'idée est de tenir compte des incertitudes sur la classification obtenue pour réaliser l'étiquetage textuel.

Nous présentons tout d'abord les métadonnées d'une image et proposons ensuite une méthode pour définir les classes de nos partitions.

8.2.1 Caractérisation d'une image à partir de métadonnées contextuelles

Une image est initialement définie par sa date de prise de vue et sa localisation. Ces informations à l'état brut sont peu pertinentes pour un utilisateur : les coordonnées GPS ou une date sous forme numérique ne sont pas utilisables telles quelles. Nous proposons d'enrichir ces métadonnées afin de fournir des mots clés avec une sémantique plus riche.

L'ajout de ces nouvelles métadonnées pour chaque image obtenue peut se faire automatiquement à partir d'une base de connaissances (pour les informations temporelles) et d'un SIG (pour les informations spatiales). Pour le cas temporel, la base de connaissances est facilement réalisée à partir de la date de chaque image. Pour les métadonnées contextuelles spatiales, des SIG sont disponibles, par exemple les systèmes Multimap ou GeoConcept.

Les tableaux 8.1 et 8.2 ci-contre présentent respectivement des exemples de métadonnées contextuelles temporelles et spatiales obtenues à partir de la date et du lieu d'une image. Chaque image est définie par plusieurs labels textuels. Le tableau comprend la probabilité *a posteriori* d'affectation des images à leur classe : nous utilisons cette probabilité dans la suite pour définir une classe. Notons que le

id	Décennie	Année	Saison	Mois	Semaine	Jours	Moment	h
1	90's	1998	été	juillet	2	jeudi	midi	1.0
2	00's	2003	hiver	décembre	3	samedi	matin	0.9
3	00's	2003	hiver	janvier	1	jeudi	matin	1.0
4	00's	2003	hiver	janvier	1	jeudi	midi	0.85

Table 8.1: Exemples de métadonnées contextuelles obtenues à partir de la date. La première colonne "id" contient les identifiants des images de la collection. La dernière colonne "h" est la probabilité a posteriori d'affectation de l'image à sa classe. Chaque image est représentée par une liste de labels textuels, récupérés à partir d'une base de connaissances. La métadonnée ***semaine*** *prend des valeurs entre* 1 *et* 5 *suivant les mois de l'année (la semaine* 1 *représente les jours du* 1 *au* 7 *dans le mois).*

id	Continent	Pays	Région	Département	Ville	Rue	h
1	Europe	France	P. de la Loire	Loire Atlantique	Nantes	rue boileau	1.0
2	Amérique	États unis	New York	*n'existe pas*	New York	Broadway Av.	1.0
3	Europe	France	P. de la Loire	Sarthe	Le Mans	rue Bollé	0.6
4	Europe	France	P. de la Loire	Sarthe	Le Mans	rue Bollé	1.0

Table 8.2: Exemple de métadonnées contextuelles obtenues à partir de la localisation. La première colonne contient les identifiants des images de la collection. La dernière colonne "h" est la probabilité a posteriori d'affectation de l'image à sa classe. Chaque image est représentée par une liste de labels textuels, récupérés à partir d'un SIG.

pré-traitement proposé sur les données spatiales (voir page 122) entraîne qu'une donnée de la classification spatiale peut représenter plusieurs images. La probabilité d'affectation *a posteriori* de cette donnée est associée à toutes les images qu'elle représente.

Dans les deux tableaux, les métadonnées contextuelles sont organisées hiérarchiquement, fournissant une classification identique à un arbre de décision (néanmoins, nous avons expliqué dans le chapitre 1 qu'une telle approche n'est pas pertinente pour parcourir une collection comprenant beaucoup d'images). Les métadonnées d'une image sont définies sur plusieurs niveaux, leur nombre dépendant de la base de connaissances et du SIG.

Soit une image i, nous définissons ses métadonnées m_i par :

$$m_i = \{m_i^1, m_i^2, \ldots, m_i^l\}, m_i^j \in M,$$

où m_i^l est un label textuel défini au niveau l, l est le nombre de niveaux pour représenter une image, dépendant de la base de connaissances et du SIG, et M l'ensemble des valeurs possibles des labels textuels. Nous notons respectivement M^t et M^s les ensembles des valeurs des labels temporelles et spatiales. Une image i est définie par un résumé temporel m_i^e et un résumé spatial m_i^s.

Dans les tableaux 8.1 et 8.2, les métadonnées temporelles et spatiales sont respectivement représentées sur des ensembles à 7 et 6 niveaux. Plus le niveau de la métadonnée est élevé, plus elle est précise : si $j < k$ alors nous pouvons en déduire que le mot clé m_i^k est une information plus précise que m_i^j.

8.2.2 Caractérisation d'une classe à partir des métadonnées contextuelles de ses images

Nous présentons une méthode pour définir une classe à partir des métadonnées contextuelles des images qu'elle contient. Nous proposons d'obtenir un *résumé* de la classe défini par :

- un ensemble de labels textuels représentant les différentes valeurs des métadonnées contextuelles temporelles et spatiales de ses images ;

- pour chaque label, le poids au sein de l'ensemble, mettant en valeur sa pertinence vis à vis de la classe. Ce poids est calculé à partir des affectations *a posteriori* des images à la classe. La prise en compte de ces probabilités va permettre de valoriser les métadonnées contextuelles des images qui sont fortement associées à la classe.

Comme nous l'avons vu dans la section précédente, les métadonnées contextuelles sont organisées hiérarchiquement. Pour que ces résumés aient un sens, une classe sera représentée seulement par les niveaux ayant des métadonnées identiques et le premier niveau comprenant plusieurs valeurs différentes. En effet, si une classe contient des images de Hollande et de France, le résumé doit contenir les valeurs {{**Europe**}, {**Hollande**, **France**} pour les niveaux *Continent* et *Pays*. Et il serait peu pertinent que le niveau *Région* contienne un ensemble de valeurs appartenant à des pays différents.

Le résumé d'une classe c de la partition temporelle ou spatiale est défini par :

$$\begin{aligned} c &= \{\langle e_1^1, \ldots, e_1^{l-1}, \{\alpha_1/e_1^l, \ldots, \alpha_r/e_r^l\}\rangle, \\ &\quad \langle s_1^1, \ldots, s_1^{l'-1}, \{\alpha_1/s_1^{l'}, \ldots, \alpha_{r'}/s_{r'}^{l'}\}\rangle\}, \end{aligned}$$

où $e_i^j \in M^e$, $s_{i'}^j \in M^s$, et l et l' sont les niveaux à partir desquels la classe contient des images avec des labels textuels différents. Au niveau l (respectivement l') des hiérarchies des métadonnées, une classe est représentée par r (respectivement r') labels ayant chacun un poids α représentant sa proportion. La notation α/e^l signifie que le label textuel e^l a un poids de α dans la classe. Nous proposons de le calculer à partir des probabilités d'affectation *a posteriori* de l'image à sa classe.

Le poids d'un label e_r^l est obtenu en sommant les probabilités des images associées à ce label, et en la divisant par la somme totale des probabilités de la classe :

$$\alpha_j = \frac{\sum_{i|e_r^l = m_i} h_i}{\sum_{i=1}^{n_k} h_i},$$

où h_i est la probabilité d'affectation de l'image à sa classe (obtenue à partir de la matrice t_{ik} des partitions temporelles et spatiales) et n_k le nombre d'images dans la classe.

Exemple de résumés de classes

Nous proposons deux exemples pour bien illustrer notre approche. Si une classe contient les images 1 et 2 des tableaux 8.1 et 8.2, le résumé est défini par :

$$\begin{aligned} m &= \{\langle\{\frac{1.0}{1.9} = 0.52/\textbf{90's}, \frac{0.9}{1.9} = 0.48/\textbf{00's}\}\rangle, \\ &\quad \langle\{0.5/\textbf{Amérique}, 0.5/\textbf{Europe}\}\rangle\} \end{aligned}$$

Nous avons ici $l = l' = 1$. Le premier niveau des métadonnées étant différent dans les deux cas, il n'est pas nécessaire de descendre plus bas. Pour une classe comprenant les images 1, 3 et 4 :

$$m = \{\langle\{1.0/2.85 = 0.35/\textbf{90's}, 1.85/2.85 = 0.65/\textbf{00's}\}\rangle, \langle\textbf{Europe}, \textbf{France}, \textbf{P. de la loire}, \{1.0/2.6 = 0.38/\textbf{Loire Atlantique}, 1.6/2.6 = 0.62/\textbf{Sarthe}\}\rangle\}$$

Notre méthode permet de faire apparaître les différentes valeurs des métadonnées comprises dans la classe. Le poids affecté à chacune d'entres elles indique la pertinence de la valeur vis à vis de la classe. La prise en compte des probabilités d'affectation des données permet de favoriser les images censées représenter au mieux l'événement ou le lieu de la classe.

Cette approche est pertinente pour fournir un système basé sur des requêtes. Nous proposons deux méthodes pour gérer ces requêtes :

- l'utilisateur pose des contraintes à l'aide de mots clés sur les partitions de la collection. Durant son parcours, seules les composantes vérifiant ces contraintes sont présentées ;
- l'utilisateur ne veut pas parcourir sa collection et trouver directement un événement précis. Les composantes vérifiant la requête peuvent être organisées suivant leur score.

Dans les deux cas, les requêtes sont réalisées sur les événements de la collection et non pas sur une image précise. Les utilisateurs recherchant leurs images par événements, cela semble plus pertinent.

8.3 Construction de partitions hybrides géo-temporelles

Nous proposons deux types de classifications hybrides, l'une basée sur des partitions temporelle et spatiale obtenues avec notre algorithme d'optimisation, et la seconde sur les classifications hiérarchiques. Dans les deux propositions, nous nous basons sur un axe temporel pour présenter les classes d'images. Cet axe est facilement interprétable par un utilisateur et une solution basée sur un ordonnancement des lieux est plus complexe. Nous pourrions utiliser une carte géographique, mais son parcours présente aussi des problèmes d'IHM et cette méthode n'est de plus pas plébiscitée par les utilisateurs [80].

8.3.1 Combinaison des partitions temporelle et spatiale non-hiérarchiques par un critère statistique

Cette approche consiste à combiner deux partitions, l'une temporelle et l'autre spatiale, afin de fournir une partition unique. Suivant les épisodes temporels, nous organiserons les images à partir de la partition temporelle ou spatiale suivant leur qualité. Afin d'évaluer la qualité des partitions, nous nous basons sur un critère de classifiabilité : l'entropie (ce critère est défini par l'équation 4.12, page 67). La figure 8.1 illustre la technique de construction de la partition hybride.

Nous notons e et s respectivement les partitions temporelle et spatiale initiales, obtenues avec notre algorithme incrémental d'optimisation. Notre combinaison des partitions temporelle et spatiale consiste à :

1. transformer la partition spatiale en une partition temporelle. Pour ce faire, nous définissons une nouvelle partition temporelle e' obtenue en divisant les classes spatiales en sous-classes de données

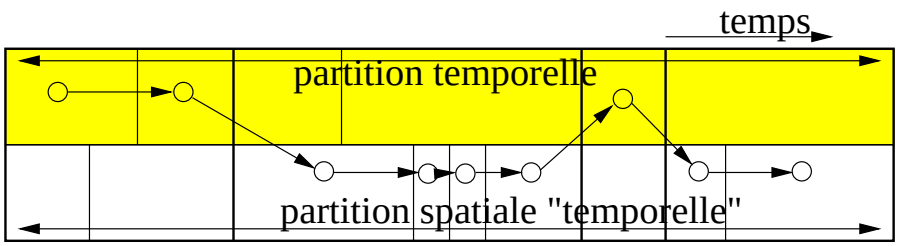


Figure 8.1: Illustration de la technique de combinaison de partitions temporelle et spatiale par un critère statistique. La partition temporelle initiale est en grise et celle en dessous représente une partition spatiale projetée sur un axe temporel. Pour les deux partitions, les lignes représentent les limites de classes. La partition temporelle et la partition spatiale divisée en sous-ensembles temporels sont comparées sur des intervalles de temps avec des limites de partitions communes (lignes en gras). Les flèches représentent la partition hybride obtenue. Le choix entre les deux partitions est basé sur le critère d'entropie, permettant de choisir la partition de meilleure qualité.

temporellement continues. Pratiquement cela revient à parcourir les images suivant leur date de prise de vue et à délimiter une classe à chaque fois que deux images successives sont dans des classes spatiales différentes ;

2. détecter les intervalles de temps $\triangle(e, e')$ de e et e' qui présentent des limites de classes identiques, c'est à dire, rechercher les limites communes entre les deux partitions (les instants où un changement de lieu et un changement d'événement coïncident) ;

3. pour chaque intervalle $\triangle(e, e')$, calculer les critères d'entropie des classes comprises dans $\triangle(e, e')$ pour les partitions e et e', en se basant respectivement sur les probabilités *a posteriori* des partitions initiales temporelles et spatiales. Nous sélectionnons ensuite sur l'intervalle $\triangle(e, e')$ la partition minimisant ce critère (nous rappelons qu'une partition a une entropie faible si elle est composée de classes distinctes avec des données bien regroupées). De cette manière, pour chaque intervalle, nous sélectionnons la partition la plus pertinente pour parcourir la collection.

Notons que le calcul des deux critères d'entropie est réalisé à partir des partitions initiales temporelle et spatiale (à partir des matrices d'affectations associées à chacune des partitions). Nous comparons la qualité de la partition temporelle à la partition spatiale (et non pas à la partition spatiale projetée sur l'axe temporel). Dans le cas où la partition spatiale a une meilleure entropie sur un intervalle donné, nous la sélectionnons : l'utilisateur parcourt la partition spatiale *transformée* où chaque classe est liée à un lieu de la collection.

L'algorithme 6 ci-contre détaille notre technique de combinaison. Nous rappelons que la matrice d'affectation t est composée de n lignes (n est le nombre de données) et K colonnes (K est le nombre de classes). Chaque terme $t_{ik} \in [0, 1]$ est la probabilité *a posteriori* que la donnée i soit générée par la composante k. Le critère d'entropie d'une composante k se calcule à partir de ces termes :

$$E_k = -\sum_{i=1}^{n} t_{ik} \cdot ln(t_{ik}) \tag{8.1}$$

Algorithme 6 Construction d'une partition hybride non hiérarchique

en entrée : une partition temporelle e et une partition spatiale s, définies sur les mêmes images. Nous notons t^e et t^s respectivement les matrices d'affectation *à posteriori* des données aux composantes de e et s. Nous notons I_i l'image i de la collection ;
1.création d'une partition temporelle e' à partir de s :
pour i allant de 1 à n **faire**
  si $I_i \in s_j, 1 \leq j \leq K^s$ et $I_{i+1} \notin s_j$ **alors**
    nous posons une limite de classe entre I_i et I_{i+1} dans la classification e' ;
  finsi
fin pour
2.sélectionner les intervalles de temps minimaux $\triangle(e, e')$ tels que les limites entre les classes temporelles de e et e' coïncident ;
pour pour chaque intervalle $i \in \triangle(e, e')$ **faire**
  3.1.calculer l'entropie E_e du groupe de classes temporelles de e sur i, à partir de la matrice des t^e ;
  3.2.calculer l'entropie E_s du groupe de classes temporelles de e' sur i, à partir de la matrice des t^s (seulement à partir des données incluses dans i);
  si $E_e < E_s$ **alors**
    3.3.sélectionner la classification temporelle sur l'intervalle i ;
  sinon
    3.4.sélectionner la classification spatiale sur l'intervalle i ;
  finsi
fin pour

La combinaison des deux partitions est ainsi réalisée en prenant en compte la qualité des partitions en se basant sur le critère d'entropie. Nous rappelons néanmoins que ce critère est optimisé dans le cas où la partition n'est composée que d'une seule classe. Si sur un intervalle une des deux partitions ne comprend qu'une seule classe avec une entropie très faible, il y a de fortes chances qu'elle soit choisie.

L'avantage essentiel de cette approche est qu'elle permet de fournir une partition de meilleure qualité que les partitions initialement obtenues puisque nous sélectionnons les meilleures parties de chacune. Par contre elle ne fait plus la distinction entre un changement d'événements temporels ou de lieux entre les classes et elle est peu pertinente pour parcourir une large collection : pour un grand nombre d'images, le nombre de classes peut être très élevé. Nous la proposons juste pour parcourir les images récentes de l'utilisateur, celles-ci étant susceptibles d'être les plus recherchées.

8.3.2 Combinaison des partitions temporelles et spatiales hiérarchiques

Nous proposons maintenant une classification hybride basée sur des classifications hiérarchiques temporelle et spatiale.

La conception d'une IHM efficace pour parcourir un arbre sur un appareil mobile est complexe. Ce type de structure est difficile à représenter et nécessite de nombreuses interactions pour parcourir les noeuds dans différents sens (en largeur ou en profondeur). De plus, dans notre cas, nous avons à disposition deux classifications hiérarchiques, ce qui rend le problème encore plus compliqué. La figure 8.2 ci-après détaille toutes les possibilités de parcours à partir d'un noeud de l'arbre temporel ou spatial. À partir d'un noeud temporel, il est possible de :

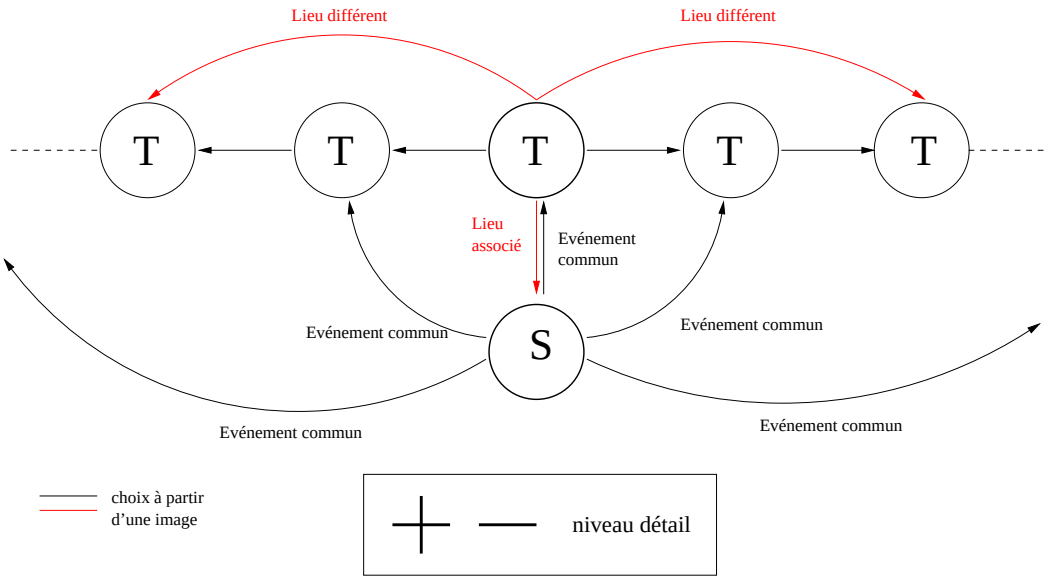


Figure 8.2: Graphe détaillant toutes les possibilités pour parcourir une collection à partir d'une classe des classifications hiérarchiques temporelle ou spatiale.

- descendre/monter dans l'arbre suivant le niveau de détail voulu ;
- parcourir les événements suivants/précédents à un même niveau ;
- aller directement à l'événement suivant/précédent présentant un lieu différent ;
- retrouver les images associées aux différents lieux de l'événement.

À partir d'un noeud spatial, l'alternative est de :

- descendre/monter dans l'arbre suivant le niveau de détail voulu ;
- retrouver les événements suivants/précédents comprenant les lieux du noeud.

Le nombre de choix pour parcourir la collection est ainsi élevé. Fournir toutes ces alternatives à un utilisateur semble difficile en particulier sur un appareil mobile. Notre objectif est de fournir une IHM plus simplifiée, en limitant les possibilités pour parcourir la collection.

Combinaison des classes temporelles et spatiales

Notre combinaison des deux classifications hiérarchiques consiste à proposer une hiérarchie de partitions hybrides. Les classes spatiales et temporelles sont combinées comme suit:

- pour une classe temporelle donnée, nous la divisons en fonction des classes spatiales associées à ses images. Il est nécessaire de sélectionner les différentes classes spatiales qui représentent les

lieux de ses images ;

- Une classe spatiale est sélectionnée si tous ses fils contiennent au moins une image de la classe temporelle (sélection des feuilles de l'arbre spatial si un tel noeud n'existe pas). Cela revient à rechercher les composantes spatiales les plus générales pour représenter au mieux les images de la classe temporelle. Une fois les composantes spatiales sélectionnées, nous les fusionnons en nous basant sur leurs métadonnées contextuelles (voir ci-après) ;

- la composante temporelle est représentée par plusieurs classes spatiales. L'utilisateur parcourt les classes temporelles et a la possibilité pour chacune d'entres elles de voir directement les lieux que les composent. Ainsi, il peut choisir directement dans une classe temporelle, le lieu qui l'intéresse.

La division d'une classe temporelle avec les classes spatiales qui la composent est définie à partir des métadonnées contextuelles de ces dernières. Après avoir sélectionné les classes spatiales, nous les définissons à partir de leurs métadonnées contextuelles comme explicité dans la section 8.2 et nous les fusionnons en réalisant les étapes suivantes :

- sélectionner le niveau l le plus général dans la hiérarchie des métadonnées tel que leurs labels textuels inter-classes diffèrent ;

- regrouper les classes ayant des métadonnées identiques au niveau l dans une même classe.

Par exemple, si une composante temporelle contient les composantes spatiales suivantes :

$$\begin{aligned} m_1 &= \{\langle \textbf{Europe}, \textbf{France}, \textbf{Pays de la loire}, \{\textbf{Loire Atlantique}, \textbf{Sarthe}\}\rangle\} \\ m_2 &= \{\langle \textbf{Amérique du Nord}, \{\textbf{États Unis}, \textbf{Canada}\}\rangle\} \\ m_3 &= \{\langle \textbf{Amérique du Nord}, \textbf{États Unis}, \{\textbf{New York}, \textbf{Vermont}\}\rangle\} \end{aligned}$$

Alors, cette classe temporelle sera définie spatialement par les deux classes {**Europe**} et {**Amérique du Nord**}. Les valeurs des labels sont différentes pour le niveau *continent* et les deux composantes définies par l'Amérique du Nord sont regroupées ensemble. Chaque événement est bien défini par les différents lieux qui le composent.

L'algorithme 7 page 152 explicite en détail notre algorithme de construction de la partition hiérarchique hybride.

Exemple d'une partition hiérarchique hybride à partir de la collection réelle

La figure 8.3 page 155 présente un exemple d'une partition hybride obtenue à partir de la collection réelle. Sur l'axe temporel, nous avons accès aux événements temporels de la collection, et pour chacun d'entre eux, les différents lieux les caractérisant. L'utilisateur parcourt sa collection, et pour chaque classe choisie, pose des contraintes simultanément sur un événement et un lieu. Une partition hybride est ensuite reconstruite en les prenant en compte. Notons que l'interface présentée est seulement un exemple pour comprendre notre algorithme. Les classes sont résumées à l'aide de leurs métadonnées contextuelles mais nous pouvons aussi envisager de les résumer à partir d'images représentatives de chaque événement (tout en limitant le nombre d'images à afficher). Le choix des images *représentatives* reste encore à définir.

Algorithme 7 Construction d'une partition hybride hiérarchique

initialisation : nous notons s et e respectivement les partitions temporelle et spatiale.
q^e est le noeud temporel courant de la partition e initialisé avec la racine de l'arbre.
L'ensemble d'images $\mathcal{I}$ est initialisé avec les images contenues dans q^e (toutes les images de la collection).
Nous rappelons que c_q est l'ensemble des fils du noeud q.
pour chaque noeud $q_i \in c_{q^e}$ tel qu'il existe au moins une image $I \in \mathcal{I}$ **faire**
2.1.sélectionner dans l'arbre s les noeuds $q^s \in s$ tels que pour tout noeud $q' \in c_{q^s}$, il existe au moins une image $i \in \mathcal{I} \cap q' \cap q_i$;
2.2.construire le résumé spatial de chaque noeud q^s, seulement à partir des images $i \in \mathcal{I} \cap q^s$ (le résumé ne prend pas en compte les images de q^s non incluses dans $\mathcal{I}$) ;
2.3.sélectionner le niveau l de la hiérarchie des métadonnées à partir duquel les résumés spatiaux diffèrent et regrouper les résumés avec une valeur similaire sur le niveau l ;
2.4.diviser le noeud q_i suivant le nombre de résumés r obtenus ;
fin pour
3.l'utilisateur choisit un résumé r d'un événement $q \in c_{q^e}$: $q^e = q$ et mettre à jour l'ensemble $\mathcal{I}$ à partir des images comprises dans l'ensemble choisi par l'utilisateur (les images du noeud q représentées par r) ;
4.retourner à l'étape 2.

Avantage d'une telle approche

Simplicité de l'IHM pour un appareil mobile

Le premier avantage de cette approche est la combinaison de la recherche temporelle/spatiale permettant de retrouver simplement une image. Pour chaque classe choisie, l'utilisateur fait un choix simultané sur un événement et un lieu de la collection : à chaque étape, les possibilités des classes à parcourir diminuent rapidement. Les interactions sont simplifiées pour parcourir la collection : nous avons une seule partition hiérarchique à explorer et les images peuvent être retrouvées rapidemment.

Gain en visibilité grâce à une sémantique plus riche

On peut penser que la méthode pour diviser les événements temporels en différents lieux est similaire à une approche basée sur les métadonnées. Les classes avec des métadonnées contextuelles de même valeurs (sur un niveau donné) sont regroupées : le procédé serait identique à une approche avec un arbre de décision. Néanmoins dans notre cas, les classes sont obtenues à partir d'une approche statistique et leur définition, selon notre méthode (section 8.2), permet de lier les différentes valeurs des métadonnées, selon leur similarité temporelle ou spatiale. Par exemple, sur la figure 8.3 page 155, nous obtenons une classe étiquetée *[Pays de la Loire, Bretagne]*, regroupant les valeurs de métadonnées de deux régions voisines. Un tel résultat n'est pas possible si l'on se base seulement sur une approche basée sur les métadonnées contextuelles. Un point essentiel de notre approche est ainsi un gain en visibilité grâce à une sémantique plus riche des métadonnées, tout en gardant les avantages de son caractère statistique.

8.4 Expérience : construction de partitions hybrides non hiérarchiques

Nous avons appliqué l'algorithme 6 page 149 sur les partitions obtenues avec notre algorithme d'optimisation avec les collections artificielle réaliste et réelle, utilisées dans les expériences des chapitres précédents.

Collection artificielle réaliste

La partition spatiale obtenue avec notre algorithme d'optimisation est présentée au chapitre 5, page 99. Nous la transformons en une partition temporelle pour la comparer avec la partition temporelle initiale.

La figure 8.4 page 156 présente la partition hybride construite à partir de la collection artificielle réaliste. La partition temporelle obtenue précédemment avec notre algorithme d'optimisation est détaillée par la figure (a). La partition spatiale projetée sur un axe temporel est représentée par la figure (b). Les intervalles de temps présentant des limites de classes similaires sont mis en valeur grâce aux doubles flèches entres les deux figures. Nous obtenons 4 intervalles de temps à comparer.

L'algorithme consiste à choisir pour chaque intervalle la meilleure partition (entre un intervalle de la figure (a) et (b)) à présenter à l'utilisateur en se basant sur leur entropie. Les flèches indiquent le parcours de la partition hybride et mettent en valeur les intervalles sélectionnés. Les 2 premiers intervalles sont choisis sur la partition spatiale, puis les 2 derniers sont sélectionnés sur la partition temporelle. Ce résultat est pertinent puisque les données temporelles sur les deux premiers intervalles sont mal structurées. La partition obtenue sur cette partie est composée de beaucoup de classes proches. Son entropie est supérieure à celle de la partition spatiale car les composantes de la partition spatiale sont bien définies et présentent une faible entropie. Pour les derniers intervalles, le choix se porte sur la partition temporelle, les classes y étant clairement définies.

Collection réelle de Guillaume B.

La figure 8.5 page 157 présente la partition hybride obtenue à partir de la collection réelle de Guillaume B.. Les partitions temporelle et spatiale obtenues avec notre algorithme d'optimisation sont présentées au chapitre 5, respectivement page 102 et page 104.

Le premier intervalle est sélectionné sur la partition temporelle. Ce cas est intéressant puisqu'il met en valeur le défaut du critère d'entropie. En effet, la partition temporelle sur cet intervalle est composée d'une seule classe : le critère d'entropie est donc maximisé si cette classe présente une entropie nulle. Étant dans un tel cas, c'est donc la partition temporelle qui est choisie ici. Nous aurions préféré que ce soit la partition spatiale puisqu'elle aurait fourni plus de détails.

Les 3 intervalles suivants sont choisis sur la partition spatiale. Ce choix semble pertinent au vu de la partition temporelle qui comprend plusieurs composantes discutables par rapport aux classes distinctes de la partition spatiale. La partition hybride revient ensuite sur la partition temporelle pour 5 intervalles. La partition temporelle est ici de bonne qualité, présentant juste quelques sur-segmentations. Il faut noter que le critère d'entropie n'est pas pertinent pour détecter ce type de défaut. Une sur-segmentation entraîne en général une baisse de l'entropie.

Enfin, le choix se reporte sur la partition spatiale pour le $10^{\text{ème}}$ intervalle alors que les données temporelles sont convenablement classées. Ce résultat est néanmoins cohérent puisque nous obtenions une partition spatiale présentant des composantes avec une faible entropie.

Pour les derniers intervalles, la partition hybride passe à la partition temporelle, puis à la partition spatiale et revient à la partition temporelle. Pour des raisons de clarté, cela n'est pas montré sur la figure.

Discussion

Dans les deux cas présentés, notre algorithme semble efficace pour choisir la partition la plus pertinente. Nous avons vu que quand la partition temporelle était mal définie, le choix tendait vers la partition spatiale. Dans les deux cas, les partitions spatiales obtenues avec notre algorithme d'optimisation étaient pertinentes, ce qui explique que quand la partition temporelle présentait une entropie élevée, la partition spatiale était préférée. Pour que la technique soit efficace, il faut que au moins une des deux partitions soit de bonne qualité sinon nous n'obtenons par d'amélioration.

8.5 Conclusion

Nous avons tout d'abord présenté dans ce chapitre une technique pour résumer les composantes d'un modèle de mélange à partir de ces métadonnées. Cette représentation des événements ou des lieux nécessite de définir une base de connaissances et d'un SIG. Le résumé des classes à partir de métadonnées contextuelles est adéquat aux contraintes liées aux terminaux mobiles : l'affichage de texte est peu coûteux et prend moins de place, comparé avec des résumés obtenus à l'aide d'images.

Ensuite nous avons proposé deux techniques pour construire des partitions hybrides spatio-temporelles. La première proposition est basée sur des partitions temporelle et spatiale obtenues avec notre algorithme d'optimisation. La partition hybride est une combinaison des deux partitions construites en sélectionnant, à l'aide du critère d'entropie, les intervalles temporels de meilleure qualité.

La deuxième proposition est basée sur les classifications hiérarchiques temporelle et spatiale. Nous avons proposé de combiner les deux partitions en un graphe généré dynamiquement en fonction des requêtes de l'utilisateur. Les avantages d'une telle approche sont :

- l'efficacité de la recherche et l'IHM appropriée pour un appareil mobile ;
- le gain en visibilité grâce à une sémantique plus riche.

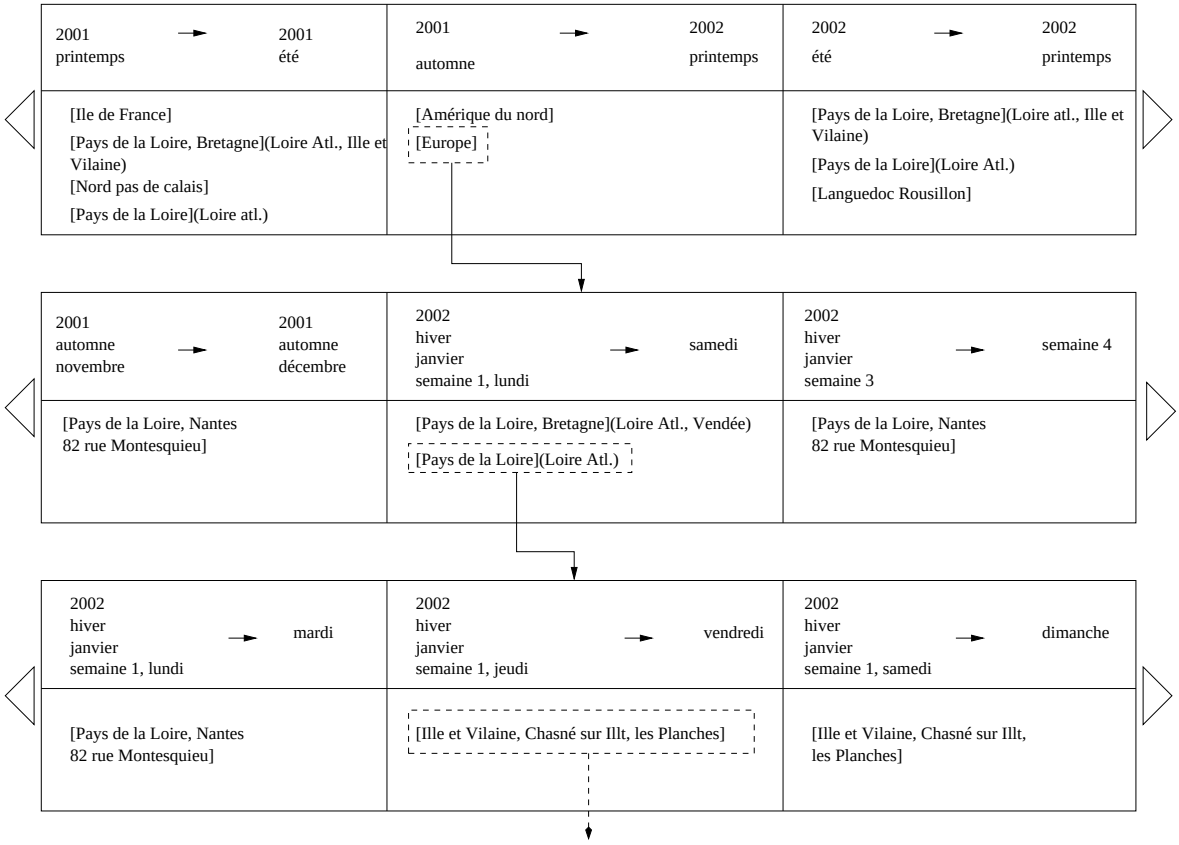


Figure 8.3: Exemple de classification hiérarchique hybride obtenue à partir de classifications hiérarchiques temporelle et spatiale. Les flèches symbolisent le choix de l'utilisateur. Le premier niveau montre la partition la plus générale : elle est logiquement composée de 5 *composantes (la racine de l'arbre obtenu précédemment contient* 5 *composantes, voir la figure 7.8, page 140) mais nous n'en montrons que* 3*. Les indications des lieux pour ces* 3 *événements sont obtenues à partir de la classification hiérarchique spatiale initiale. Pour la deuxième classe du premier niveau, définie spatialement par* ***[Amérique du Nord]*** *et* ***[Europe]****, l'événement temporel comprenait plus de deux classes spatiales. Mais comme la différence entre les métadonnées se situait au niveau des continents, nous ne présentons que ces métadonnées (voir l'exemple précédent où les deux classes* Amérique du Nord *sont fusionnées). Nous avons noté qu'une telle division n'existait pas dans la classification spatiale initiale. La combinaison des deux classifications à l'aide de notre algorithme permet ainsi d'améliorer la partition spatiale. Nous obtenions directement un niveau trop détaillé pour un utilisateur qui peut souhaiter avoir tout d'abord des divisions par continent et ensuite par pays. Le deuxième niveau est généré à partir du choix de l'utilisateur : il décide de rechercher des images en Europe entre l'automne 2001 et le printemps 2002. Les classes présentées dans le niveau* 2 *sont des événements ne comprenant que des lieux inclus en Europe. L'intervalle de temps est automatiquement mis à jour : il commence en automne 2001 pour finir en hiver 2002 (les images datées du printemps 2002 ont été prises en Amérique du Nord). L'utilisateur fait ensuite le choix de parcourir les images prises pendant la première semaine de janvier en Pays de la Loire. Le dernier niveau est défini sur un intervalle d'une semaine sur deux lieux distincts. Nous arrivons très vite à un bon niveau de détail dans la collection.*

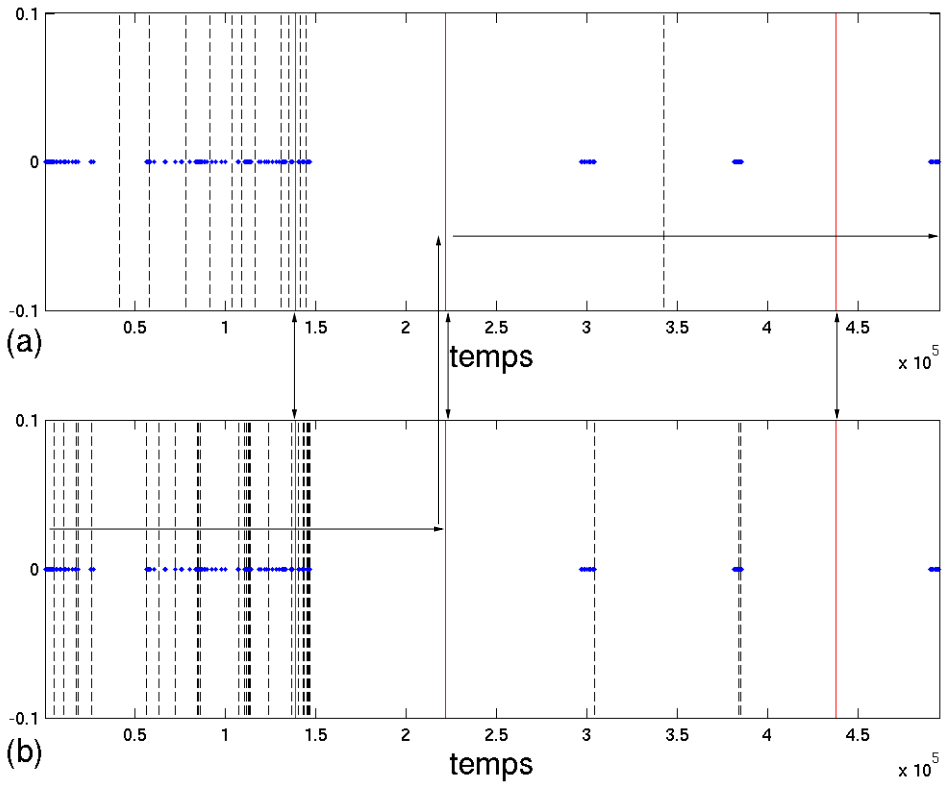


Figure 8.4: Collection réaliste : partition hybride. La figure (a) est la partition temporelle et la figure (b) la projection de la partition spatiale. Les traits en pointillés sont les limites entre les classes. Quand les limites sont similaires pour les deux classifications, le trait est plein. Les doubles flèches entres les figures permettent de mieux les mettre en valeur. Les flèches indiquent les intervalles sélectionnés pour parcourir la collection (les intervalles minimisant le critère d'entropie). Les 2 premiers intervalles sont choisis sur la partition spatiale, puis les 2 derniers sont sélectionnés sur la partition temporelle. Ce résultat est pertinent puisque les données temporelles sur les deux premiers intervalles sont mal structurées. La partition obtenue sur cette partie est composée de beaucoup de classes proches. Son entropie est supérieure à celle de la partition spatiale car les composantes de la partition spatiale sont bien définies. Pour les derniers intervalles, le choix se porte sur la partition temporelle, les classes y étant clairement définies.

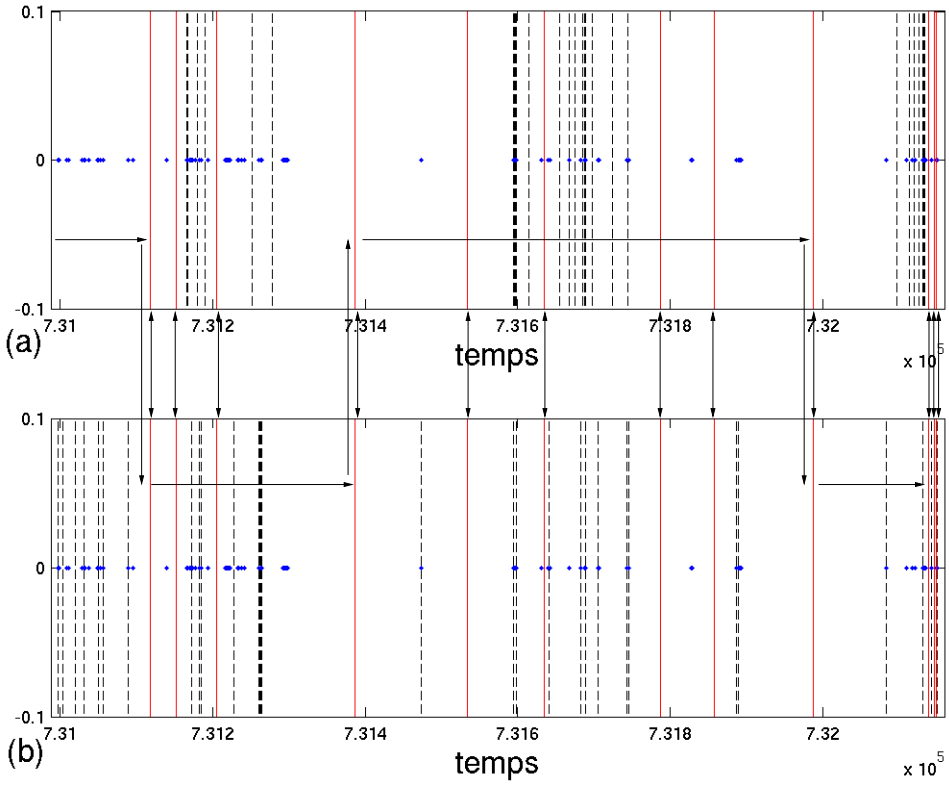


Figure 8.5: Collection réelle : partition hybride. La figure (a) est la partition temporelle et la figure (b) la projection de la partition spatiale. Les traits en pointillés sont les limites entre les classes. Quand les limites sont similaires pour les deux classifications, le trait est plein. Les doubles flèches entres les figures permettent de mieux les mettre en valeur. Les flèches indiquent les intervalles sélectionnés pour parcourir la collection (les intervalles minimisant le critère d'entropie). Le premier intervalle est sélectionné sur la partition temporelle : cela est dû à la propriété du critère d'entropie qui est optimisé quand une partition ne contient qu'une composante (cas de la partition temporelle sur le premier intervalle). Les 3 *intervalles suivants sont choisis sur la partition spatiale, ce qui semble pertinent au vu de la partition temporelle qui comprend plusieurs composantes discutables. Ensuite, la partition hybride revient sur la partition temporelle pour* 5 *intervalles. Ce résultat est cohérent car elle semble de bonne qualité. Enfin, le choix se reporte sur la partition spatiale pour le* $10^{\text{ème}}$ *intervalle alors que les données temporelles sont convenablement classées. La partition spatiale présentant des composantes avec une faible entropie, ce résultat est néanmoins cohérent. Pour les derniers intervalles, la partition hybride passe à la partition temporelle, puis à la partition spatiale et revient à la partition temporelle. Pour des raisons de clarté, cela n'est pas montré sur la figure.*

Conclusion

Dans cette thèse, nous avons proposé plusieurs techniques de structuration d'une collection d'images personnelles acquises à partir d'un terminal mobile. Notre contribution est la conception d'algorithme incrémental fournissant des partitions adaptées pour les parcourir sur ce type d'appareil. Notre processus de construction des classifications dépend peu de paramètres critiques à régler manuellement.

Tout d'abord, nous avons présenté les propriétés des collections d'images personnelles et les systèmes d'organisation existants. Les collections personnelles présentent des différences avec des collections *standards* (collection d'images de sources diverses) motivant la nécessité de développer de nouveaux outils pour les organiser : les critères de recherches, la connaissance partielle de la collection de l'utilisateur et les nouvelles métadonnées sont des éléments à prendre en compte pour fournir une technique adéquate. Les principales solutions proposées dans la littérature sont pour le moment liées à des paramètres arbitraires et peu proposent la possibilité d'obtenir des résumés de la collection. Notre travaux ont en partie consisté à améliorer ces points.

Nous avons choisi de traiter la structuration de la collection d'images comme un problème de classification. Notre approche est basée sur la construction de deux partitions distinctes, l'une temporelle et l'autre spatiale, à partir des métadonnées des images : leur date et leur géolocalisation. Ces choix sont motivés par la disponibilité des métadonnées et leur pertinence vis à vis des critères favoris des utilisateurs [95]. Nous avons conçu et mis en oeuvre un algorithme de classification qui présente les deux avantages suivants :

- il est incrémental : les partitions peuvent être mises à jour pour chaque nouvelle image en n'ayant généralement que des modifications mineures à effectuer. Cet aspect est important au vu de la faible autonomie en énergie des appareils mobiles ;

- il est hiérarchique : des résumés de la collection sont construit afin de faciliter son parcours sur le terminal mobile.

Nous avons choisi de modéliser nos données à l'aide d'une approche probabiliste. Les modèles de mélange présentent des avantages pour fournir un système incrémental (affectation des données aux classes susceptibles d'évoluer) et automatique (existence d'algorithmes d'estimation des paramètres de complexité raisonnable et de critères statistiques pour les comparer). Les principaux ingrédients de notre approche sont les modèles de mélange gaussien, dont les paramètres sont estimés à l'aide d'une adaptation de l'algorithme EM, et le critère statistique ICL pour comparer les partitions obtenues.

Nos premiers travaux ont proposé un algorithme incrémental d'optimisation du critère ICL pour construire deux partitions non hiérarchiques, temporelle et spatiale. Nous avons adapté les travaux de Ueda [115] pour concevoir notre algorithme. L'approche est basée sur des sauts semi-locaux dans l'espace des paramètres (des fusions et divisions de composantes) pour trouver un bon optimum du critère ICL. Le système présente l'avantage d'être automatique, les partitions étant déterminées à partir de la structure des données. Les partitions obtenues lors de nos expériences nous semblent correctement représenter la structure des données. Elles évoluent convenablement en fonction de l'ajout de nouvelles images.

Ensuite, nous avons proposé un algorithme hiérarchique incrémental afin de fournir des résumés de la collection. Cette approche est basée sur la combinaison de notre algorithme d'optimisation et l'algorithme hiérarchique agglomératif de Fraley [44]. Nous avons proposé une méthode de mise à jour permettant de ne reconstruire que les parties de l'arbre affectées par la nouvelle donnée. Les avantages de notre approche sont :

- les paramètres essentiels dirigeant la modélisation sont estimés plutôt que fixés arbitrairement. La configuration de l'arbre est déterminée par la structure des données ;

- les partitions obtenues dans l'arbre sont évaluées afin de choisir celles présentant de bonnes propriétés : des classes distinctes avec des données bien regroupées. La mise à jour de l'arbre comprend une phase de sélection des niveaux pertinents basée sur le critère statistique ICL.

Enfin, nous avons présenté des méthodes pour faciliter le parcours de la collection sur des appareils mobiles. Après avoir proposé une technique pour résumer les classes d'images à l'aide de métadonnées contextuelles, nous avons introduit deux méthodes de construction de partitions hybrides. La première proposition est basée sur des partitions temporelle et spatiale obtenues avec notre algorithme d'optimisation. La classification hybride est une partition du temps en "épisodes", où la notion d'épisodes s'appuie sur la partition temporelle ou spatiale selon le degré de fiabilité de chacune. La deuxième proposition est basée sur les classifications hiérarchiques temporelle et spatiale. Nous avons proposé de combiner les deux partitions en un graphe généré dynamiquement en fonction des requêtes de l'utilisateur.

Pour résumer, les contributions de nos travaux sont les suivantes :

- un algorithme incrémental d'optimisation du critère ICL basé sur les modèles de mélange ;

- un algorithme hiérarchique et incrémental, combinant notre algorithme d'optimisation, un algorithme de sélection de niveaux et un algorithme agglomératif classique ;

- des méthodes pour faciliter le parcours de la collection sur un appareil mobile :
  - la définition des composantes à partir de métadonnées contextuelles ;
  - des partitions hybrides non-hiérarchique et hiérarchique.

Ces travaux ont donné lieu à plusieurs publications, listées page 163.

Quelques perspectives

Une perspective à court terme porte tout d'abord sur nos deux algorithmes incrémentaux, en particulier sur la méthode d'ajout des nouvelles données dans les classifications. Nous ne prenons pas totalement en compte le processus de construction de la collection : l'utilisateur a tendance à prendre des images par paquets. Une optimisation serait ainsi d'ajouter directement dans la classification des groupes d'images associées à un événement. Nous avons vu qu'en ajoutant les images une à une, l'optimisation du critère ICL peut nécessiter une recherche conséquente pour éviter de tomber dans un minimum local : au fur et à mesure que les données d'un paquet sont ajoutées, les paramètres du modèle sont optimisés localement sur ces données et il peut être plus difficile de les mettre à jour si un changement d'événements apparaît. La phase d'optimisation du critère ICL pourrait ainsi être améliorée si nous évitons toutes les

restructurations liées à l'apparition d'un nouvel événement. L'objectif consiste à détecter les changements d'événement avant de les ajouter dans la classification afin de construire les groupes limitant au maximum les modifications sur la classification. Le groupe ajouté doit bien sûr pouvoir être détaillé dans la classification si sa structure le permet. Une solution peut être d'utiliser des techniques similaires à celles présentées dans le chapitre 1, proposant de fixer des limites fixes pour détecter un changement d'événement (une limite de temps ou une distance).

Une autre perspective est de proposer une partition spatio-temporelle hiérarchique, en extension à notre partition hybride non-hiérarchique. La partition découlerait de la combinaison des arbres temporelle et spatiale, en fonction de la qualité des partitions des différents niveaux. La difficulté réside dans le choix des partitions à sélectionner (comparaison de partitions temporelle et spatiale à des niveaux de détails différents) et la manière de les combiner pour fournir une classification cohérente (un niveau de classes "spatiales" ayant des fils organisés temporellement).

A long terme, nos travaux porteront tout d'abord sur la généralisation de notre système à tous types de documents numériques disponibles sur les appareils mobiles. Notre proposition étant basée sur les métadonnées des images, elle peut être appliquée sur tous types de documents ayant ces informations disponibles. Il serait intéressant de pouvoir concevoir un véritable journal personnel à partir de toutes les données multimédia disponibles sur les terminal mobiles : texte, son, audio, vidéo et image. L'objectif est de proposer une solution d'archivage de données multimédia similaire à des applications telles que MyLifeBits [52] ou MEMEX (Microsoft Research Digital Memories). Les problèmes à résoudre sur ces systèmes sont nombreux (voir l'introduction de la thèse) et nous désirons nous focaliser en particulier sur les points suivants :

- la représentation des résumés : Dans le chapitre 8 nous proposons une représentation à l'aide des métadonnées contextuelles, qui est adaptée au vu des contraintes d'IHM sur un appareil mobile. Plusieurs solutions sont néanmoins possibles. L'utilisateur peut par exemple préférer une visualisation des images représentatives de chaque classe. Un choix pertinent des images est difficile et nécessite des traitements sur le contenu des images (détection de visages, évaluation de leur qualité, ...). Ce problème devient plus complexe quand plusieurs types de documents sont disponibles (choix et représentation des vidéos [117], des documents textuels, ...). Les objectifs à résoudre sont la sélection des documents pertinents et leur représentation dans le résumé ;

- le problème d'accessibilité : un système adéquat doit être mis en place pour permettre une disponibilité continue et rapide de l'information. La collection doit être accessible à partir de n'importe quel appareil (un ordinateur personnel ou un appareil mobile) et doit pouvoir être partagée facilement. Une architecture client-serveur mettant à disposition une base de données des images semble adaptée pour répondre à ces contraintes. Le lieu du serveur n'est pas défini mais une solution réaliste peut consister à faire appel à un hébergeur de données qui va garantir un stockage sûr et un accès permanent (le stockage du serveur directement sur un appareil mobile n'est pas réalisable pour le moment au vu de leur faible autonomie et de la capacité des mémoires). Dans le cas d'un accès via un appareil mobile (et c'est le cas le plus pertinent pour garantir une accessibilité continue, au vu de leur disponibilité), il est nécessaire de fournir une architecture du type *Client mobile - serveur fixe*. La réalisation d'une telle architecture présente de nombreux problèmes à résoudre [9] : la confidentialité des données, leur synchronisation (si les données sont partagées) ou l'adaptibilité du système par exemple (offrir à l'utilisateur un service aussi proche que possible de celui fourni dans un contexte classique).

Liste des publications de nos travaux

Revue

A. PIGEAU et M. GELGON, Organizing a personal image collection with statistical model-based ICL clustering on spatio-temporal camera phone meta-data, *Journal of Visual Communication and Image Representation*, Pages 425-445, Volume 15(3), Septembre 2004.

Internationales

A. PIGEAU et M. GELGON, Building and tracking hierarchical geographical & temporal partitions for image collection management on mobile devices, *International Conference of ACM Multimedia*, Pages 141-150, Singapour, novembre 2005.

A. PIGEAU et M. GELGON, Incremental Statistical geo-temporal structuring of a personal camera phone image collection, *17th International Conference on Pattern Recognition (ICPR'2004)*, Pages 878-881, Cambridge, United Kingdom, août 2004.

A. PIGEAU, G. RASCHIA, M GELGON, N. MOUADDIB et R. SAINT-PAUL, A fuzzy linguistic summarization technique for TV recommender systems, *IEEE International Conference of Fuzzy Systems (FUZZ-IEEE'2003)*, Pages 743-748, St-Louis, USA, mai 2003.

Nationales

A. PIGEAU et M. GELGON, Structuration géo-temporelle incrémentale hiérarchique d'une collection d'images pour sa gestion sur un mobile, à paraître dans le *Congrès Reconnaissance des Formes et Intelligence Artificielle (RFIA'2006)*, Tour, janvier 2006.

A. PIGEAU et M. GELGON, Organisation statistique spatio-temporelle d'une collection d'images acquises d'un terminal mobile géolocalisé, *Congrès Reconnaissance des Formes et Intelligence Artificielle (RFIA'2004)*, Pages 76-84, LAAS/Toulouse, janvier 2004.

Worshop et Symposium

A. PIGEAU et M. GELGON, Hybrid spatio-temporal structuring and browsing of an image collection acquired from a personal camera phone, *ISIVC2004 proceedings, second international Symposium on Image/Video Communications over fixed and mobile networks*, Pages 25-28, Brest, France, juillet 2004.

A. PIGEAU et M. GELGON, Spatio-temporal organization of one's personal image collection with model-based ICL-clustering, *3rd International Workshop on Content-Based Multimedia Indexing (CBMI'2003)*, Pages 111-118, Rennes, France, septembre 2003.

Divers

A. PIGEAU et M. GELGON, Collections multimedia personnelles sur les mobiles : problèmes & perspectives, *Journée du GDR-ISIS "Les applications de l'indexation multimédia: mythes ou réalités"*, janvier 2003.

Bibliographie

[1] R. AGRAWAL, J. GEHRKE, D. GUNOPULOS et P. RAGHAVAN. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 94–105, juin 1998.

[2] H. AKAIKE. Information theory as an extension of the maximum likelihood principle. In B. PETROV et F. CSAKI, réds., *Second international Symposium on Information Theory*, pages 267–281, 1973.

[3] H. AKAIKE. A new look at the statistical identification model. In *IEEE Transactions on Automatic Control*, volume 19, pages 716–723, 1974.

[4] H. AKAIKE. Information Measures and Model Selection. *Bulletin of the International Statistical Institute*, 50:277–290, 1983.

[5] D. ASHBROOK et T. STARNER. Learning Significant Locations and Predicting User Movement with GPS. In *IEEE International Symposium on Wearable Computing (ISWC'2002), Seattle, USA*, pages 101–108, octobre 2002.

[6] M. BALABANOVIC, L. L. CHU et G. J. WOLFF. Storytelling with digital photographs. In *Proceedings of Human Factors in Computing Systems (CHI'2000) ACM Press, The Hague, The Netherlands*, pages 564–571, avril 2000.

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

[8] P. BERKHIN. Survey Of Clustering Data Mining Techniques. Rapport technique, Accrue Software, San Jose, CA, 2002.

[9] G. BERNARD, J. BEN-OTHMAN, L. BOUGANIM, G. CANALS, B. DEFUDE, J. FERRIÉ, S.GANÇARSKI, R. GUERRAOUI, P. MOLLI, P. PUCHERAL, C. RONCANCIO, P. SERRANO-ALVARADO et P. VALDURIEZ. Mobile Databases: a Report on Open Issues and Research Directions. *ACM SIGMOD Record*, 33(2):78–83, juin 2002.

[10] J. D. BEZDECK. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.

[11] C. BIERNACKI. *Choix de modèles en classification*. Thèse de Doctorat, Université de Technologie de Compiègne, 1997.

[12] C. BIERNACKI, G. CELEUX et G. GOVAERT. Assessing a Mixture Model for Clustering with the Integrated Classification Likelihood. In *IEEE Transaction on pattern analysis and machine intelligence*, volume 22, pages 719–725, juillet 2000.

[13] C. BIERNACKI, G. CELEUX et G. GOVAERT. Strategies for getting the highest likelihood in mixture models. Rapport technique No 4255, INRIA, Rhone-Alpes, Grenoble, septembre 2001.

[14] C. BIERNACKI, G. CELEUX et G. GOVAERT. Strategies for getting the highest likelihood in mixture models. Rapport technique, INRIA, 2001.

[15] C. M. BISHOP et M. SVENSEN. Robust Bayesian Mixture Modelling. In *Proceedings Twelfth European Symposium on Artificial Neural Networks*, pages 69–74, Bruges, Belgium, avril 2004.

[16] N. BOUGUILA, D. ZIOU et J. VAILLANCOURT. Novel Mixtures Based on the Dirichlet Distribution: Application to Data and Image Classification. *Lecture Notes in Computer Science*, 2734:172–181, août 2003.

[17] M. BOUTELL et J. LUO. Photo classification by integrating image content and camera metadata. In *Proceedings of International Conference on Pattern Recognition*, volume 4, pages 901–904, Cambridge, U.K., août 2004.

[18] H. BOZDOGAN. *Multi-Sample Cluster Analysis and Approaches to Validity Studies in Clustering Individuals*. Thèse de Doctorat, Department of Mathematics, University of Illinois, Chicago, 1981.

[19] H. BOZDOGAN. Determining the number of clusters in the standart multivariate normal mixture model using model selection criteria. Rapport technique, Department of Mathematics, University of Illinois, Chicago, juin 1983.

[20] H. BOZDOGAN. On the Information-Based Measure of Covariance Complexity and Its Application to the Evaluation of Multivariate Linear Models. *Communications in Statistics, Theory and Methods*, 19(1):221–278, 1990.

[21] H. BOZDOGAN. Choosing the Number of Component Clusters in the Mixture-Model Using a New Informational Complexity Criterion of the Inverse-Fisher Information Matrix. *Studies in Classification, Data Analysis, and Knowledge Organization, Springer-Verlag, Heidelberg*, pages 40–54, 1993.

[22] H. BOZDOGAN. Mixture-Model Cluster Analysis Using Model Selection Criteria and a New Informational Measure of Complexity. In *Proceedings of the First US/Japan Conference on the Frontiers of Statistical Modeling*, volume 2, pages 69–113, 1994.

[23] P. G. BRYANT. Large-Sample Results for Optimization Based Clustering Methods. *Journal of Classification*, 8:31–44, 1991.

[24] V. BUSH. As We May Think. *The Atlantic Monthly*, 176:101–108, juillet 1945.

[25] C. CELEUX et J. DIELBOLT. The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational Statistics Quarterly*, 2(1):73–92, 1985.

[26] C. CELEUX et J. DIELBOLT. L'algorithme SEM: Un algorithme d'apprentissage probabiliste pour la reconnaissance de mélange de densités. *Revue de Statistique Appliquée*, 34(2):35–52, 1986.

[27] C. CELEUX et J. DIELBOLT. Une version de type recuit simulé de l'algorithme EM. *Notes aux comptes rendus de l'académie des Sciences*, 310:119–124, 1990.

[28] C. CELEUX et G. GOVAERT. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistic and Data Analysis*, 14:315–332, 1992.

[29] G. CELEUX et G. GOVAERT. Gaussian Parsimonious Models. *Pattern Recognition*, 28(5):781–793, mai 1995.

[30] G. CELEUX et G. SOROMENHO. An entropy criterion for assessing the number of clusters in a mixture model. *Journal of Classification*, 13(2):195–212, 1996.

[31] C.-Y. CHEN, S.-C. HWANG et Y.-J. OYANG. An Incremental Hierarchical Data Clustering Algorithm Based on Gravity Theory. In *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, volume 2336, pages 237 – 250, mai 2002.

[32] M. COOPER, J. FOOTE, A. GIRGENSOHN et L. WILCOX. Temporal Event Clustering for Digital Photo Collections. In *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, volume 1, pages 269–288, août 2005.

[33] A. CUTLER et M. P. WINDHAM. Information-Based Validity Functionals for Mixture Analysis. In H. BOZDOGAN, réd., *Proceedings of the first US-Japan Conference on the Frontiers of Statistical Modeling*, pages 149–170, Amsterdam, 1993.

[34] D. CUTTING, D. KARGER, J. PEDERSEN et J. TUKEY. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings 15th Annual International ACM SIGIR Conference on Research Development in Information Retrieval*, pages 318–329, Copenhagen, Denmark, juin 1992.

[35] M. DAVIS et R. SARVAS. Mobile Media Metadata for Mobile Imaging. In *IEEE International Conference on Multimedia and Expo (ICME 2004) Special Session on Mobile Imaging*, pages 1707–1710, juin 2004.

[36] P. A. DAVOINE, J. GENSEL et H. MARTIN. A Web-based Multimedia Framework for Diffusing Spatio-Temporal Information: Application to Natural Hazards. In *12th International Conference on Geoinformatics*, pages 149–156, juin 2004.

[37] A. P. DEMPSTER, N. M. LAID et D. B. RUBIN. Maximum likelihood for incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.

[38] G. M. DJUKNIC et R. E. RICHTON. Geolocation and assisted GPS. *IEEE Computer Magazine*, 34(2):123–125, février 2001.

[39] S. T. DUMAIS, E. CUTRELL, J. J. CADIZ, G. JANCKE, R. SARIN et D. C. ROBBINS. Stuff I've Seen: A system for personal information retrieval and re-use. In *the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 72 – 79, juillet 2003.

[40] B. EFRON et R. J. TIBSHIRANI. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

[41] F. ESPINOZA, P. PERSSON, A. SANDIN, H. NYSTROM, E. CACCIATORE et M. BYLUND. GeoNotes: Social and Navigational Aspects of Location-Based Information Systems. In *Proceedings of Third International Conference on Ubiquitous Computing (Ubicomp 2001)*, pages 2–17, Atlanta, Georgia, septembre 2001.

[42] M. ESTER, H. P. KRIEGEL, J. SANDER et X. XU. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the second International Conference on Knowledge Discovery and Data Mining, (KDD)*, pages 226–231, août 1996.

[43] R. A. FISHER. *Statistical methods and scientific inferences*. Edinburgh : Oliver and Boyd, 1956.

[44] C. FRALEY. Algorithms for Model-Based Gaussian Hierarchical Clustering. *SIAM Journal on Scientific Computing*, 20(1):270–281, 1999.

[45] C. FRALEY et A. E. RAFTERY. Bayesian Regularization for Normal Mixture Estimation and Model-Based Clustering. Rapport technique 486, Department of Statistics, University of Washington, Seattle, août 2005.

[46] E. FREEMAN et D. GELERNTER. Lifestreams: A Storage Model for Personal Data. *SIGMOD Record*, 25(1):80–86, mars 1996.

[47] J. F. FRIEDMAN. Regularized discriminant analysis. *Journal of the Royal Statistical Society*, 84:17–42, 1989.

[48] U. GARGI. Modeling and clustering of photo capture streams. In *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, pages 47 – 54, novembre 2003.

[49] U. GARGI, Y. DENG et D. R. TRETTER. Managing and Searching Personal Photo Collections. Rapport technique HPL-2002-67, HP Laboratories, Palo Alto, mars 2002.

[50] M. GELGON. Using face detection for browsing personal slow video in a small terminal and worn camera context. In *IEEE International Conference on Image Processing (ICIP'2001)*, pages 1062–1065, Thessaloniki, Greece, septembre 2001.

[51] M. GELGON et K. TILHOU.. Structuring the personal multimedia collection of a mobile device user based on geolocation. In *IEEE International conference on Multimedia and Expo (ICME'2002)*, pages 248–252, Lausanne, Switzerland, août 2002.

[52] J. GEMMELL, R. LUEDER et G. BELL. The MylifeBits lifetime store. In *Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence*, pages 80–83, novembre 2003.

[53] J. GOLDBERGER et S. ROWEIS. Hierarchical Clustering of a Mixture Model. In *Neural Information Processing Systems 17 (NIPS)*, pages 505–512, décembre 2005.

[54] G. H. GOLUB et C. F. V. LOAN. *Matrix Computations*. The Johns Hopkins University Press, octobre 1996.

[55] A. GRAHAM, H. GARCIA-MOLINA, A. PAEPCKE et T. WINOGRAD. Time as Essence for Photo Browsing Through Personal Digital Libraries. In *ACM Joint Conference on Digital Libraries JCDL*, pages 326–335, juin 2002.

[56] T. GREENE et W.S RAYENS. Covariance pooling and stabilization for classification. *Computational Statistic and Data Analysis*, 11(1):17–42, janvier 1991.

[57] N. GRIRA, M. CRUCIANU et N. BOUJEMAA. Unsupervised and semi-supervised clustering: a brief survey. Rapport technique, A Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence, 2004.

[58] P. GROS, R. FABLET et P. BOUTHEMY. *New descriptors for image and video indexing*, chapter State-of-the-Art in Content-Based Image and Video Retrieval, pages 213–234. Kluwer, 2001.

[59] J. HAN, M. KAMBER et A. K. H. TUNG. *Geographic Data Mining and Knowledge Discovery*, chapter Spatial Clustering Methods in Data Mining: A Survey, pages 1–29. Taylor and Francis, octobre 2001.

[60] A. HINNEBURG et D. A. KEIM. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In *Knowledge Discovery and Data Mining*, pages 58–65, août 1998.

[61] J. P. HOFFBECK et D. A. LANDGREBE. Covariance matrix estimation and classification with limited training data. *IEEE Transaction of Pattern Analysis and Machine Intelligence*, 18(7):763–767, juillet 1996.

[62] A. K. JAIN, M. N. MURTY et P. J. FLYNN. Data clustering: a review. *Source ACM Computing Surveys (CSUR) archive*, 31(3):264 – 323, septembre 1999.

[63] A. K. JAIN, A. P. TOPCHY, M. H. C. LAW et J. M. BUHMANN. Landscape of Clustering Algorithms. In *Proceedings of International Conference on Pattern Recognition*, volume 1, pages 260–263, Cambridge, U.K., août 2004.

[64] D. R. KARGER, K. BAKSHI, D. HUYNH, D. QUAN et V. SINHA. Haystack: A General Purpose Information Management Tool for End Users of Semistructured data. In *Conference on Innovative Data Systems Research*, pages 14–27, janvier 2005.

[65] R. E. KASS et A. E. RAFTERY. Bayes factors and model uncertainty. *Journal of the American Statistical Association*, 90:773–795, 1995.

[66] A. K. KURTZ. A research test of Rorschach test. *Personnel Psychology*, 1:41–53, 1948.

[67] E. LEVINE et E. DOMANY. Resampling Method for Unsupervised Estimation of Cluster Validity. *Neural Computation*, 13(11):2573–2593, novembre 2001.

[68] H. LINHART et W. ZUCCHINI. *Model Selection*. Wiley, New York, 1986.

[69] A. LOUI et A. E. SAVAKIS. Automatic Image Event Segmentation and Quality Screening for Albuming Applications. In *IEEE Proceedings International Conference on Multimedia and Expo (ICME'2000)*, pages 1125–1128, New York, USA, août 2000.

[70] C. E. LUNNEBORG. *Data Analysis by Resampling: Concepts and Applications*. Duxbury Press; 1st edition, décembre 1999.

[71] S. MARKS et O. J. DUNN. Discriminant functions when the covariance matrices are unequal. *Journal of the American Statistical Association*, 69(346):555–559, juin 1974.

[72] N. MARMASSE et C. SCHMANDT. Location-Aware Information Delivery with ComMotion. In *Handheld and Ubiquitous Computing HUC'2000, Second International Symposium*, pages 157–171, Bristol, UK, septembre 2000.

[73] N. MARMASSE, C. SCHMANDT et D. SPECTRE. WatchMe: Communication and Awareness Between Members of a Closely-Knit Group. In *International Conference on Ubiquitous Computing (Ubicomp)*, pages 214–231, septembre 2004.

[74] J. F. MCCARTHY et E. S. MEIDEL. ACTIVE MAP: A Visualization Tool for Location Awareness to Support Informal Interactions. In *Handheld and Ubiquitous Computing, First International Symposium, HUC'99*, volume 1707, pages 158–170, septembre 1999.

[75] B. MINAEI-BIDGOLI, A. TOPCHY et W. F. PUNCH. Ensembles of Partitions via Data Resampling. In *International Conference on Information Technology: Coding and Computing (ITCC)*, pages 188–192, Las Vegas, avril 2004.

[76] M. MOEGLEINAND et N. KRASNER. An introduction to snaptrack server-aided GPS. In *the institute of navigation conference, ION GPS*, 1998.

[77] R. MOHR, P. GROS et C. SCHMID. Efficient Matching with Invariant Local Descriptors. In *Proceedings of the Joint IAPR International Workshops SSPR98 and SPR98 : Advances in Pattern Recognition, Sydney, Australia*, volume 1451 of *Lecture Notes in Computer Science*, pages 54–71, août 1998.

[78] P. MULHEM et J. LIM. Home photo retrieval: Time matters. In *International Conference on Image and Video Retrieval, Lecture Notes in Computer Science*, volume 2728, pages 308–317. Springer, juin 2003.

[79] A. MYKA. Nokia Lifeblog - towards a truly personal multimedia information system. In *Workshop des GI-Arbeitskreises "Mobile Datenbanken und Informationssysteme"*, février 2005.

[80] M. NAAMAN, S. HARADA, Q. WANG et A. PAEPCKE. Adventures in space and time: Browsing personal collections of geo-referenced digital photographs. Rapport technique, Stanford University, Stanford USA, avril 2004.

[81] M. NAAMAN, Y. J. SONG, A. PAEPCKE et H. GARCIA-MOLINA. Automatic organization for digital photographs with geographic coordinates. In *International Conference on Digital Libraries archive Proceedings of the 2004 joint ACM/IEEE conference on Digital libraries*, pages 53–62, juin 2004.

[82] M. NAAMAN, Y. J. SONG, A. PAEPCKE et H. GARCIA-MOLINA. Automatically generating metadata for digital photographs with geographic coordinates. In *International World Wide Web Conference archive, Alternate track papers & posters of the 13th international conference on World Wide Web*, pages 244–245, mai 2004.

[83] M. NAAMAN, R. B. YEH, H. GARCIA-MOLINA et A. PAEPCKE. Leveraging Context to Resolve Identity in Photo Album. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 178–187, juin 2005.

[84] R. NEAL. Probabilistic inference using Markov chain Monte Carlo methods. Rapport technique CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

[85] R. NEAL et G. HINTON. *Learning in graphical models*, chapter A view of the EM algorithm that justifies incremental, sparse and other variants, pages 355–368. Dordrecht : Kluwer academic publishers, 1998.

[86] D. ORMONEIT et V. TRESP. Improved Gaussian Mixture Density Estimates Using Bayesian Penalty Terms and Network Averaging. In David S. TOURETZKY, Michael C. MOZER et Michael E. HASSELMO, réds., *Advances in Neural Information Processing Systems*, volume 8, pages 542–548. The MIT Press, 1996.

[87] B. W. PARKINSON. *Global Positioning System : Theory and Applications*. American Institude of Aeronautics and Astronautics, 1996.

[88] D. PEEL et G. J. MCLACHLAN. Robust mixture modelling using the *t* distribution. *Statistics and Computing*, 10(4):339–348, octobre 2000.

[89] J. C. PLATT et B. A. Field M. CZERWINSKI. PhotoTOC: Automatic Clustering for Browsing Personal Photographs. Rapport technique MSR-TR-2002-17, Microsoft Research, février 2002.

[90] J.C. PLATT. Autoalbum: Clustering Digital Photographs Using Probabilistic Model Merging. In *Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 96–100, juin 2000.

[91] M. QUENOUILLE. Approximate tests of correlation in time series. *Journal of the Royal Statistical Society*, 11:18–84, 1949.

[92] B. J. RHODES. The Wearable Remembrance Agent: A system for augmented memory. *Personal Technologies Journal Special Issue on Wearable Computing*, 1:218–224, 1997.

[93] D. RIDDER et V. FRANC. Robust subspace mixture models using t-distributions. In R. HARVEY et J.A. BANGHAM, réds., *Proceedings British Machine Vision Conference, BMVA*, pages 319–328, Norwich, septembre 2003.

[94] J. RISSANEN. Stochastic Complexity in Statistical Inquiry. *World Scientific*, 1989.

[95] K. RODDEN. How Do People Manage Their Digital Photographs? In *ACM Conference on Human Factors in Computing Systems*, pages 409 – 416, Fort Lauderdale, avril 2003.

[96] V. ROTH, T. LANGE, M. BRAUN et J. BUHMANN. A resampling approach to cluster validation. In *International Conference on Computational Statistics*, pages 123–130, juin 2002.

[97] S. SALVADOR et P. CHAN. Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 576–584, novembre 2004.

[98] R. SARVAS, E. HERRARTE, A. WILHELM et M. DAVIS. Metadata Creation System for Mobile Images. In *International Conference On Mobile Systems, Applications And Services, Proceedings*

of the 2nd international conference on Mobile systems, applications, and services, pages 36–48, juin 2004.

[99] C. SCHMID. Constructing models for content-based image retrieval. In *Proceedings IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 39–45, décembre 2001.

[100] G. SCHWARZ. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[101] B. SHNEIDERMAN et H. KANG. Direct Annotation: A Drag-and-Drop Strategy for Labeling Photos. In *IV '00: Proceedings of the International Conference on Information Visualisation*, page 88. IEEE Computer Society, juin 2000.

[102] S. SHOHAM. Robust clustering by deterministic agglomeration EM of mixtures of multivariate t-distributions. *Pattern Recognition*, 35(5):1127–1142, mai 2002.

[103] A. SMAILAGIC et R. MARTIN. Metronaut: A Wearable Computer with Sensing and Global Communication Capabilities. In *First International Symposium on Wearable Computers (ISWC 1997)*, pages 116–122, octobre 1997.

[104] A. W. M. SMEULDERS, M. WORRING, S. SANTINI, A. GUPTA et R. JAIN. Content-based image retrieval at the end of the early years. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, décembre 2000.

[105] P. SMYTH. Clustering Using Monte-Carlo Cross-Validation. In AAAI PRESS, réd., *the second International Conference on Knowledge Discovery and Data Mining*, pages 126–133, Portland, août 1996.

[106] S. S. SOLIMAN et C. E. WHEATLEY. Geolocation technologies and applications for third generation wireless. *Wireless Communication and Mobile Computing*, 2(3):229–251, mai 2002.

[107] S. TADJUDIN. *Classification of High Dimensional Data With Limited Training Samples*. Thèse de Doctorat, Purdue University, Indiana, USA, mai 1998.

[108] S. TADJUDIN et D. A. LANDGREBE. Covariance estimation with limited training samples. *IEEE Transaction on Geoscience and Remote sensing*, 37(4):134–149, juin 1999.

[109] J. TANG, N. YANKELOVICH, J. BEGOLE, M. VANKLEEK, F. LI et J. BHALODIA. ConNexus to Awarenex: Extending awareness to mobile users. In *Proceedings of the CHI'01 Conference on Human Factors in Computing Systems*, pages 221–228, mars 2001.

[110] J. TANTRUM, A. MURUA et W. STUETZLE. Hierarchical model-based clustering of large datasets through fractionation and refractionation. *Information Systems*, 29(4):315–326, juin 2004.

[111] C. E. THOMAZ et D. F. GILLIES. small sample size: a methodological problem in bayes plug-in classifier for image recognition. Rapport technique, Department of Computing, Imperial College of Science Technology and Medicine, London, juin 2001.

[112] R. TIBSHIRANI, G. WALTHER, D. BOTSTEIN et P. BROWN. Cluster validation by prediction strength. Rapport technique, Statistics Department,, Stanford University, 2001.

[113] R. TIBSHIRANI, G. WALTHER et T. HASTIE. Estimating the number of clusters in a dataset via the Gap statistic. In *Journal of the Royal Statistical Society: : Series B (Statistical Methodology)*, volume 63, pages 411–423, 2001.

[114] K. TOYAMA, R. LOGAN, A. ROSEWAY et P. ANANDAN. Geographic location tags on digital images. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 156–166, Berkeley, CA, USA, novembre 2003.

[115] N. UEDA, R. NAKANO, Z. GHARHAMANI et G. HINTON. SMEM algorithm for mixture models. *Neural computation*, 12(9):2109–2128, septembre 2000.

[116] K. T. VASKO et H. T. T. TOIVONEN. Estimating the number of segments in time series data using permutation tests. In IEEE Computer Science PRESS, réd., *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 466–473, décembre 2002.

[117] J. VERMAAK, P. PÉREZ, M. GANGNET et A. BLAKE. Rapid Summarisation and Browsing of Video Sequences. In *British Machine Vision Conference, BMVC'02*, volume 1, pages 424–433, Cardiff, UK, mai 2002.

[118] P. VIOLA et M. JONES. Robust Real-time Object Detection. *International Journal of Computer Vision*, 57(2):137–154, mai 2004.

[119] W. WAGENAAR. My memory : a study of autobiographical memory over six years. *Cognitive psychology*, 18:225–252, 1986.

[120] P. W. WAHL et R. A. KRONMALL. Discriminant functions when the covariance are equal and sample sizes are moderate. *Biometrics*, 33:479–484, septembre 1977.

[121] J. H. WARD. Hierarchical groupings to optimize an objective function. *Journal of the American Statistical Association*, 58:234–244, 1963.

[122] A. WILHELM, Y. TAKHTEYEV, R. SARVAS, N. Van HOUSE et M. DAVIS. Photo annotation on a camera phone. In *Proceedings of ACM Computer Human Interaction (CHI'2004)*, pages 234–238, Vienna, Austria, avril 2004.

[123] M. P. WINDHAM et A. CUTLER. Information Ratios for Validating Cluster Analysis. *Journal of the American Statistical Association*, 87:1188–1192, 1992.

[124] C. WU. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11:95–103, 1983.

[125] T. ZHANG, R. RAMAKRISHNAN et M. LIVNY. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *the ACM-SIGMOD International Conference on Management of Data (SIGMOD-96)*, pages 103–114, juin 1996.

[126] Y. ZHAO. Standardization of Mobile Phone Positioning for 3G Systems. *IEEE Communications Magazine*, pages 108–116, juin 2002.

[127] Z.ZIVKOVIC et F.van der HEIJDEN. Recursive unsupervised learning of finite mixture models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 26(5):651–656, mai 2004.

Table des matières

Structuration géo-temporelle de données multimédia personnelles en vue de la navigation sur un appareil mobile

Antoine PIGEAU

Résumé

Les travaux de recherche présentés dans cette thèse portent sur l'organisation de collections multimédia personnelles acquises par un appareil mobile. Ce type de données est désormais de plus en plus présent dans la vie courante avec l'apparition d'appareils photographiques numériques, de téléphones mobiles équipés de capteur photographique ou encore de caméras numériques. Le problème posé est ainsi la recherche et l'indexation de ces collections afin de faciliter leur exploration future. Dans notre travail, nous nous sommes focalisés sur la classification de collections d'images personnelles acquises à partir de capteurs photographiques intégrés dans un téléphone portable.

Nous avons choisi de traiter la structuration de la collection d'images comme un problème de classification. Notre approche est basée sur la construction de deux partitions distinctes, l'une temporelle et l'autre spatiale, à partir des métadonnées des images : leur date et leur géolocalisation. Les principaux ingrédients de notre approche sont les modèles de mélange gaussien, dont les paramètres sont estimés avec une adaptation de l'algorithme EM, et le critère statistique ICL pour déterminer la complexité des modèles.

Un algorithme incrémental d'optimisation du critère ICL est tout d'abord proposé, permettant la construction de partitions non-hiérarchiques de manière automatique. Il est ensuite combiné avec un algorithme agglomératif pour fournir un algorithme hiérarchique incrémental, afin de pouvoir concevoir des résumés de la collection. Enfin nous proposons plusieurs techniques, combinant les partitions obtenues, pour construire des partitions hybrides spatio-temporelles, prenant en compte les contraintes d'IHM sur un appareil mobile.

Mots-clés : Recherche d'images, Application sur des terminaux mobiles, Métadonnées temporelles et spatiales, Classification statistique

Abstract

Usage of mobile devices raises the need for organizing large personal multimedia collection. Their permanent availability and the ability to easily share retrieved pictures make this context propitious for building large collection of multimedia data. The present work focus on personal image collections acquired from mobile phones equipped with a camera. Our objective is to provide a classification of such collection in order to simplify the browsing task on a mobile device.

We deal with the structuring of an image collection as a clustering problem. Our solution consists in building two distinct temporal and spatial partitions, based on the temporal and spatial metadata of each image. The main ingredients of our approach are the Gaussian mixture models, of which parameters are estimated with an adaptation of the EM algorithm, and the ICL criterion to determine the models complexities.

First, we propose an incremental optimization algorithm to build non-hierarchical partitions in an automatic manner. It is then combined with an agglomerative algorithm to provide an incremental hierarchical algorithm, in order to summarize the collection. Finally, two techniques are proposed, combining the partitions obtained, to build hybrid spatio-temporal classifications taking into account the human machine interaction constraints.

Keywords: Image retrieval, Mobile applications, Spatio-temporal metadata, Statistical clustering