Contributions en classification automatique : agregation bayesienne de melanges de lois et visualisation interactive

Pierrick Bruneau

UNIVERSITÉ DE NANTES

ÉCOLE DOCTORALE

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE MATHÉMATIQUES

Année : 2010

Thèse de Doctorat de l'Université de Nantes

Spécialité : Informatique

présentée et soutenue publiquement par

Pierrick BRUNEAU

le 10 décembre 2010

à l'Université de Nantes

TITRE :
Contributions en classification automatique : agrégation bayésienne de mélanges de lois et visualisation interactive

Jury

Rapporteurs :	M. Gilles Venturini	Professeur, Université François Rabelais, Tours
	M. Younès Bennani	Professeur, Université Paris 13
Examinateur :	M. Frank Nielsen	Professeur, École Polytechnique
Directeur de thèse :	M. Marc Gelgon	Professeur, Université de Nantes
Co-encadrant :	M. Fabien Picarougne	Maitre de Conférences, Université de Nantes

Composante de rattachement du directeur de thèse : Polytech'Nantes

Thèse financée par le projet ANR Safimage
Equipes GRIM et COD du LINA (UMR CNRS 6241)
EPI INRIA Atlas

N° ED 503

Contributions en classification automatique, agrégation de mélanges de lois et visualisation interactive
Résumé

Internet, ainsi que des architectures récentes telles que les réseaux de capteurs, sont le siège de masses de sources de données distribuées à large échelle, en perpétuelle croissance. Cette profusion, accompagnée du besoin d'outillage des utilisateurs, implique le développement de techniques d'analyse et d'indexation adaptées. Les techniques de classification automatique concernent la caractérisation de classes dans un ensemble d'éléments. Celles-ci sont très souvent employées pour la réalisation d'outils rendant l'information accessible aux utilisateurs. Dans le cadre de cette thèse, nous avons contribué à l'agrégation de modèles de mélange de distributions de probabilité. Cette classe de modèles est en effet souvent utilisée pour des tâches de catégorisation. Nos propositions, utilisant un formalisme bayésien variationnel, sont caractérisées par des coûts de calcul et de transmission réduits. Avec ces travaux, nous entendions fournir une solution partielle à l'estimation de modèles sur des données distribuées. Nous avons également contribué à la classification visuelle de données en flux. Pour ce faire, nous avons notamment employé des principes bio-mimétiques, ou encore des résultats de la théorie des graphes. Outre la proposition d'interfaces efficaces pour un utilisateur, nous avons également envisagé la manière dont celui-ci peut rétro-agir sur le processus de classification.
Mots-clés : classification, clustering, agrégation, modèles de mélange, classification visuelle, rétro-action utilisateur, données distribuées, estimation bayésienne

Contributions to categorization learning, mixture models aggregation and interactive visualization
Abstract

The internet and recent architectures such as sensor networks are currently witnessing tremendous and continuously growing amounts of data, often distributed on large scales. Combined with user expectations with respect to tooling, this encourages developing adequate techniques for analyzing and indexing. Classification and clustering tasks are about characterizing classes within data collections. These are often used as building blocks for designing tools aimed at making data accessible to users. In this document, we describe our contributions to mixture models aggregation. These models are classically used for content categorization. Using variational Bayesian principles, we aimed at designing low computation and transmission costs algorithms. Doing so, we aimed at proposing a building block for distributed density model estimation. We also contributed to visual classification applied to data streams. To this purpose, we employed bio-mimetic principles, and results from graph theory. More specifically, visual and dynamic abstractions of an underlying clustering process were proposed. We strived to provide users with efficient interfaces, while allowing using their actions as a feedback.
Keywords : classification, clustering, aggregation, mixture models, visual classification, user feedback, distributed data, Bayesian learning

Remerciements

Je commencerai par remercier tout spécialement mes encadrants de thèse, Marc Gelgon et Fabien Picarougne. Leur soutien, ainsi que leur disponibilité sans faille, m'ont été précieux. Nos passionnantes discussions scientifiques, ainsi que la grande liberté et confiance qu'ils m'ont accordées, ont constitué le terreau sur lequel ce travail est fondé. Outre les aspects purement scientifiques, ils ont su me guider dans la compréhension du métier de chercheur, et me transmettre leur enthousiasme. Je ne les remercierai jamais assez de cette énergie dépensée, et espère leur avoir montré qu'elle avait été employée à bon escient.

Ma gratitude ira ensuite à mes rapporteurs, Gilles Venturini et Younès Bennani, pour le temps qu'ils m'ont consacré. Grâce à leurs relectures, ils ont judicieusement pointé quelques limites et manquements de mon travail, que j'ai tâché de corriger au mieux dans cette version de manuscrit. Je remercie également Frank Nielsen, qui, outre son rôle d'examinateur, m'a fait l'honneur de présider ce jury. J'ai particulièrement apprécié sa mise en perspective et ses questions d'ouverture qui m'ont fait envisager de nombreuses extensions à ce travail.

Mes remerciements iront ensuite aux membres des équipes GRIM et COD du LINA à Polytech'Nantes, Plus spécialement, un grand merci à Pascale Kuntz, pour avoir assuré l'interim de la direction de mon début de thèse, pour des conversations toujours enrichissantes, et des encouragements qui venaient au bon moment. Merci aussi à Sylvie Leroux, pour m'avoir très souvent sauvé face à des tracas administratifs. Je remercie également tous ceux qui m'ont supporté quotidiennement dans le bureau D201 pendant ces quelques années : en particulier Quang-Khai, Ali, Daniel et Bleuenn. De manière plus générale, merci aux doctorants et aux permanents que je côtoyais régulièrement. C'est en effet une chance d'avoir pu travailler dans une ambiance aussi sereine et agréable.

Je voudrais également remercier le département informatique de Polytech'Nantes, et tout particulièrement Corinne et Leila. Leur réactivité et leur bonne humeur me manqueront désormais.

Enfin, mes derniers remerciements, mais non les moindres, vont à ma famille, à mes amis, et surtout à Fanny. Leur patience face à mon humeur versatile due à mes obscurs problèmes de scientifique forcent l'admiration. Et la patience de la petite Léna, qui a attendu que papa soutienne sa thèse pour venir au monde ce 19 décembre, a également été appréciée !

Table des matières

Chapitre 1

Introduction générale

1.1 Motivation

Les usages récents d'Internet, parfois résumés sous le concept de Web participatif, ou Web 2.0 [O'Reilly, 2005], ont conduit à une explosion de la quantité de documents disponibles, de types toujours plus variés. Outre les diffuseurs traditionnels (e.g. sites institutionnels, d'entreprises, de journaux ou de chaines de télévision), tout utilisateur peut désormais très facilement produire et diffuser ses propres contenus. Ainsi, le nombre d'entrées de Wikipédia, et la richesse de celles-ci, augmentent sans cesse. Des sites comme Youtube, Flickr, Picasa, MySpace ou last.fm permettent de diffuser très facilement nos documents multimédia. Les nombreux CMS disponibles (*Content Management System*, e.g. Joomla, WordPress) affranchissent les contributeurs de nécessaires connaissances techniques pour la mise en ligne de blogs.

Au sein de cette masse de données, en perpétuelle croissance, les utilisateurs ont besoin d'outils pour accéder efficacement à l'information souhaitée. Parmi ceux-ci, on trouve classiquement :

- **La recherche / indexation** : dans la lignée de Google pour l'indexation de pages web, sont souvent proposées des techniques recherchant les réponses les plus pertinentes à des requêtes d'utilisateurs, parmi un corpus de données disponibles. Cela peut consister à rechercher de manière plus sémantique que le célèbre moteur de recherche [Haase et al., 2009] ; ou permettre la restitution des contenus multimédia les plus *ressemblants* à un contenu *requête* soumis par l'utilisateur [Fergus et al., 2009, Jégou et al., 2010, Baluja and Covell, 2010].

- **Filtrage collaboratif** : les réseaux sociaux comme Facebook, Twitter, ou last.fm peuvent par exemple recueillir facilement les préférences de leurs utilisateurs. Des sites commerciaux comme Amazon ou PriceMinister disposent de l'historique des achats de leurs visiteurs. Le filtrage collaboratif consiste à exploiter ces informations pour estimer les préférences des utilisateurs, et prédire des sujets qui pourraient les intéresser, ou des articles qu'ils seraient plus susceptibles d'acheter.

- **La navigation / fouille de données visuelle** : une vaste collection de données brutes est difficile à envisager pour un humain ; aussi, les techniques permettant à un utilisateur de naviguer confortablement et efficacement dans une telle collection sont activement étudiées par la communauté [Huyhn and Karger, 2009, Camargo et al., 2010]. Historiquement, Yahoo a exploré cette voie en proposant

une hiérarchie de catégories pour accéder aux pages indexées, parallèlement à un moteur de recherche classique.

En parallèle de cette création continue de connaissances, les ressources humaines disponibles pour leur traitement sont toujours très limitées et coûteuses. Aussi, pour la construction des modèles employés à la réalisation des tâches exposées ci-dessus, les techniques d'apprentissage automatique sont nécessaires.

1.1.1 Classification

L'apprentissage automatique peut être défini comme l'extraction automatique d'une connaissance ou d'un modèle à partir d'une collection d'éléments de données (e.g. enregistrements de bases de données, pages web, images, vidéos, etc).

Dans ce document, nous nous intéresserons tout particulièrement à l'apprentissage de **fonctions de classification**. Une fonction de classification (ou parfois classifieur, dans la littérature) associe une classe symbolique (ou étiquette) à chaque objet d'une collection. Partant de cette définition générale, il est possible d'y associer plusieurs scénarios, éventuellement complémentaires :

- **Classification supervisée** : la fonction peut être apprise en optimisant un critère (e.g. d'erreur) relativement à un échantillon d'apprentissage formé d'éléments de données et de leurs classes respectives.

- **Classification non-supervisée** : une fonction de classification peut être déterminée en disposant d'un échantillon d'apprentissage constitué des seuls éléments de données, sans leurs classes respectives. Le critère de coût optimisé sera adapté à ce contexte. En particulier, sans les classes réelles, il ne sera pas possible de minimiser une erreur : alternativement, une bonne séparation des classes apprises pourra être recherchée. Notons dans ce cas que le nombre de classes réelles, et leur signification, sont alors inconnus a priori.

- **Validation** : la qualité des fonctions apprises peut ensuite être évaluée en utilisant un échantillon de validation. La capacité de généralisation à des éléments inconnus lors de l'apprentissage est ainsi estimée. Dans le cas spécifique de la classification non-supervisée, les classes réelles, si elles sont connues, peuvent également être employées a posteriori à fins de comparaisons avec celles découvertes par la méthode d'apprentissage. Des critères objectifs (e.g. bonne séparation des classes, nombre de groupes minimal) peuvent être également employés.

Après validation, une fonction de classification peut être utilisée afin de prédire la classe d'un nouvel élément lui étant présenté. Toutefois, en tant que telle, elle constitue une connaissance : à partir d'un échantillon brut et volumineux, souvent difficile à envisager de manière synthétique pour un être humain, une représentation compacte est

construite. Ainsi, cette représentation pourrait servir d'abstraction d'une collection de données complexes, ou d'interface avec un utilisateur humain.

Dans ce document, nous traiterons essentiellement de la classification de données numériques. En particulier, les exemples présentés seront le plus souvent issus de la littérature associée. Ce mode de représentation d'une collection d'éléments de données ne saurait cependant être général ; les questions de représentation seront brièvement abordées dans la section 2.2 afin de clairement situer le contexte de nos travaux.

1.1.2 Besoins en classification

Résumés d'information

De nouvelles architectures matérielles et logicielles, comme les réseaux pair-à-pair, les grilles de calcul, les réseaux de capteurs, ou les entrepôts de données, ont récemment émergé. Nous avons également vu que l'infiltration d'internet dans le mode de vie a engendré un contexte applicatif totalement nouveau. Toutefois, les problèmes de recherche d'information et d'analyse de données, étudiés classiquement, restent fondamentalement les mêmes. Par contre, certaines solutions classiques sont désormais inapplicables en l'état, et doivent être adaptées en conséquence.

Nous avions évoqué précédemment la limite d'humains pour le traitement d'informations : de même, l'exploitation centralisée de telles sources de données devient de plus en plus difficile, à la fois en raison de coûts de transmission, de traitement, ou encore de problématiques liées à l'anonymat et au respect de la vie privée [Coull et al., 2009, Ghinita et al., 2009]. La prise en compte de ces aspects est essentielle dans le contexte actuel, et est très souvent étudiée dans la littérature récente.

La manière la plus immédiate de limiter la quantité d'informations à transmettre sur un réseau est de réduire leur dimensionalité. Cette forme de compression peut être réalisée de manière classique avec des techniques comme l'ACP (*Analyse en Composantes Principales*). Cette question est toujours d'actualité, par exemple dans le contexte de vidéos acquises par un système distribué, où la dimensionalité des données à transmettre est réduite au moyen de projections aléatoires [Sulic et al., 2010].

Une autre approche consiste à transmettre des paramètres de modèles probabilistes sur le réseau, en lieu et place des données brutes [Gu, 2008, Safarinejadian et al., 2010].

La diffusion et l'agrégation de modèles de classification, ou autre résumé de données, sur un réseau de sources (ou dépôts) de données s'accompagne de stratégies variables pour la construction d'une information globale. Certains auteurs réalisent la mise à jour de chaque noeud en fusionnant l'information locale avec les modèles reçus [Gu, 2008, Safarinejadian et al., 2010]. D'autres ont défini une stratégie pour la construction et la diffusion distribuée d'un modèle s'améliorant progressivement au fil des noeuds parcourus [Nikseresht and Gelgon, 2008]. Parfois, les modèles diffusés et reçus sont uti-

lisés pour la mise à jour de représentations ou d'index hiérarchiques [Vasconcelos, 2001, Bechchi et al., 2007].

Flux de données

Les méthodes classiques sont adaptées au traitement d'un échantillon statique, alors que de nombreux besoins applicatifs sont associés à des données présentées en flux (e.g. réseau, capteurs). La classification des données doit donc être effectuée de manière incrémentale, soit en adaptant les méthodes classiques à ce nouveau contexte, soit en proposant de nouvelles approches. L'adaptation en ligne, et en temps réel, de méthodes d'apprentissage classique est également une préoccupation actuelle de la communauté de recherche [Seidl et al., 2009, Li and Fei-Fei, 2010].

De même, quand classiquement les prédictions ne portaient que sur un nombre figé de classes, beaucoup de travaux actuels tendent à supprimer ces restrictions : par exemple en permettant la découverte de classes inconnues dans un système de classification ou de recherche dans une collection de contenus [Liu et al., 2010, Nikitidis et al., 2010], ou, dans le cas particulier d'un contexte non-supervisé, en permettant l'ajustement automatique du nombre de groupes adéquat à la nature des données. Les méthodes bayésiennes, qui seront étudiées de manière extensive dans ce document, se prêtent particulièrement bien à ce contexte (voir section 3.5.1).

Interaction et visualisation

Le point de vue classique assimilait également l'utilisateur à un expert devant paramétrer les algorithmes d'apprentissage. Nombre de travaux récents tentent de présenter l'utilisateur non comme un expert a priori, mais comme un agent dans le cadre d'un apprentissage adaptif. En supposant que l'utilisateur puisse visualiser en temps réel l'état d'un processus de classification, celui-ci pourrait effectuer des retours (*feedback*) de diverses manières, par exemple en indiquant dynamiquement ses préférences (e.g. évolution du nombre de classes), ou en indiquant au système ce qu'il considère comme des erreurs [Ono et al., 2009, Aihara and Takasu, 2010]. Les techniques de classification visuelle (voir chapitre 5) constituent un support idéal pour ce scénario.

1.2 Contributions

Dans le cadre de cette thèse, nous avons tâché de proposer de nouvelles solutions pour l'adéquation aux scénarios d'utilisations évoqués ci-dessus. En particulier, l'adaptation de méthodes d'apprentissage statistique au contexte de **données distribuées** a été explorée. La nature changeante des flux d'information réels a été considérée, en

étudiant des **adaptations incrémentales** à des méthodes existantes. Des adaptations de méthodes de classification visuelle ont également été explorées, de manière à pouvoir considérer un acteur humain comme un agent d'influence dynamique, et non comme un expert prescripteur comme cela est effectué classiquement.

Dans un premier temps, nous proposons un **état de l'art du domaine de la classification automatique**. Après avoir évoqué les différentes natures que peuvent avoir les données à traiter, et spécifié le cadre de nos travaux, nous illustrons la richesse des solutions possibles pour l'estimation de modèles de classification, en évoquant les approches statistique, bio-mimétique, ou encore spectrale. Avec l'approche semi-supervisée, nous évoquons une solution intermédiaire à la dichotomie classique entre présence et absence de supervision. Le contexte technologique actuel, caractérisé par une production continue de documents en tous les points d'Internet, est enfin considéré, avec des apports sur la classification incrémentale, ou opérant sur des données distribuées. Des approches visuelles à la classification sont également présentées. Celles-ci constituent en effet un support intéressant pour l'interaction avec un utilisateur.

Nous proposons ensuite un état de l'art plus détaillé d'un domaine spécifique : **l'apprentissage bayésien de modèles de mélanges probabilistes**, une forme fonctionnelle classiquement utilisée en tant que fonction de classification. Après avoir rappelé les principes et outils élémentaires de l'apprentissage statistique, nous introduisons les modèles de mélange, et les dilemmes qui y sont classiquement liés. Nous effectuons ensuite une étude extensive des **algorithmes variationnels**. Leurs propriétés théoriques sont détaillées, ainsi que l'application sur les mélanges de gaussiennes et d'ACP probabilistes. Des résultats expérimentaux appuient l'intérêt de cette dernière approche. Le traitement variationnel du mélange d'ACP probabilistes souffrait de lacunes dans la littérature existante, aussi nous avons synthétisé et complété les contributions existantes [Bruneau et al., 2010b,c].

Une présentation détaillée de l'approche variationnelle se justifie en ce qu'elle a inspiré nombre de nos contributions. En particulier, nous l'avons utilisée pour proposer une **méthode d'agrégation parcimonieuse de mélanges de gaussiennes**, opérant sur les seuls paramètres des modèles à agréger [Bruneau et al., 2008a, 2010a]. Cette technique a notamment été appliquée à la structuration géo-temporelle d'une collection d'images personnelles [Bruneau et al., 2008c]. De manière plus générale, elle constitue une solution efficace à l'estimation distribuée de densités de probabilité. Nous avons proposé une variante semi-supervisée de cette technique [Bruneau et al., 2009a, 2010d]. Le modèle utilisé permet en effet de prendre en compte l'origine des composantes fusionnées, au lieu de réaliser ce qui équivaut à une réduction de mélange de gaussiennes redondant. Enfin, nous avons proposé une méthode d'**agrégation de mélanges d'ACP probabilistes** [Bruneau et al., 2010b,c]. Ce modèle permet une réduction de dimensionalité locale à chaque composante, qualité que nous avons transposé à leur agrégation. En outre, de nombreux détails théoriques, algorithmiques et d'implémentation originaux sont exposés.

Nous traiterons ensuite davantage de la cohabitation entre des approches visuelles à la classification et les scénarios d'utilisation évoqués plus haut. Ainsi, une méthode de classification visuelle, **d'inspiration bio-mimétique**, basée sur le mouvement d'agents dans un espace 2D, sert de fondement à notre contribution suivante. Celle-ci a été adapté au **traitement incrémental** d'un flux de données, constituant ainsi un outil de suivi efficace par un utilisateur [Bruneau et al., 2008b]. En particulier, la quantité et la complexité des données est abstraite par une population d'agents visualisée qui reste stable, plus adaptée à la cognition de l'utilisateur. La prise en compte d'étiquetages effectués manuellement a également été proposée [Bruneau et al., 2009b]. En effet, l'environnement 2D constitue une interface intéressante pour l'enregistrement d'entrées de la part d'un agent humain ; l'algorithme de classification visuelle est adapté en conséquence.

Enfin, nous proposons un système original pour la **visualisation du processus de classification d'une collection d'images** [Bruneau et al., 2010e,f]. Notre système considère en particulier le contexte de données en flux. Nous proposons d'abord un modèle de représentation adapté pour les images, et un algorithme de classification incrémentale. L'espace de représentation des données et de la classification est difficile à envisager pour un humain, aussi nous décrivons un système graphique réalisant une métaphore visuelle de ce processus complexe. Les groupes d'images, et leurs centroïdes respectifs, sont ainsi utilisés pour la construction d'un graphe, ensuite plongé en 3D. Ce graphe suit dynamiquement l'évolution du processus de classification grâce à un algorithme classique force et ressort. Celui-ci repose sur l'application de principes physiques, proposant ainsi des évolutions faciles à interpréter pour le système visuel humain. De plus, la visualisation constitue le point d'entrée idéal pour les rétro-actions d'un utilisateur. En interprétant celles-ci, une supervision peut être associée au processus de classification sous-jacent.

1.3 Liste des publications au 30.09.2010

Revues internationales

- P. Bruneau, F. Picarougne, and M. Gelgon. **Interactive unsupervised classication and visualization for browsing an image collection**. *Pattern Recognition*, 43(2):485–493, 2010e
- P. Bruneau, M. Gelgon, and F. Picarougne. **Parsimonious reduction of Gaussian mixture models with a variational-Bayes approach**. *Pattern Recognition*, 43:850–858, Mars 2010a

Conférences Internationales

- P. Bruneau, M. Gelgon, and F. Picarougne. **Aggregation of probabilistic PCA mixtures with a variational-Bayes technique over parameters**. *Actes de IEEE/IAPR International Conference on Pattern Recognition (ICPR'10)*, Istanbul (Turquie), Août 2010c
- P. Bruneau, F. Picarougne, and M. Gelgon. **Incremental semi-supervised clustering in a data stream with a flock of agents**. *Actes de IEEE Congress on Evolutionary Computing (CEC'2009)*, Trondheim (Norvège), Mai 2009b
- P. Bruneau, M. Gelgon, and F. Picarougne. **Parsimonious variational-Bayes mixture aggregation with a Poisson prior**. *Actes de European Signal Processing Conference (EUSIPCO'2009)*, Glasgow (UK), Août 2009a
- P. Bruneau, A. Pigeau, M. Gelgon, and F. Picarougne. **Geo-temporal structuring of a personal image database with two-level variational-Bayes mixture estimation**. *LNCS 5811 - Revised selected papers from Adaptive Multimedia Retrieval Workshops (AMR'08-AMR'09)*, pages 127–139, 2008c
- P. Bruneau, M. Gelgon, and F. Picarougne. **Parameter-Based Reduction of Gaussian Mixture Models with a Variational-Bayes Approach**. *Actes de IEEE/IAPR International Conference on Pattern Recognition (ICPR'08)*, Tampa (USA), Décembre 2008a

Conférences Nationales

- P. Bruneau, F. Picarougne, and M. Gelgon. **Visualisation dynamique de classification d'images semi-supervisée incrémentale**. *Actes de joint meeting of the Société Francophone de Classification and the Classification and Data Analysis Group of the Italian Society of Statistics (SFC-CLADAG'10)*, La Réunion (France) 2010f
- P. Bruneau, M. Gelgon, and F. Picarougne. **Fusion bayésienne de mélanges de gaussiennes par une approche variationnelle**. *Actes de Reconnaissance de Formes et Intelligence Artificielle (RFIA'10)*, nommé parmi les 5 meilleurs articles, Caen (France), Janvier 2010d
- P. Bruneau, F. Picarougne, and M. Gelgon. **Incremental clustering in a data stream with a flock of agents**. *Actes de joint meeting of the Société Francophone de Classification and the Classification and Data Analysis Group of the Italian Society of Statistics (SFC-CLADAG'08)*, Naples (Italie), Juin 2008b
- M. Gelgon, A. Maillet, P. Bruneau, F. Picarougne, and R. Lehn. **The Safim@ge Platform**. *Actes de Network and Electronic Media Forum (NEM'2008)*, Paris (France) 2008

Divers

- P. Bruneau, M. Gelgon, and F. Picarougne. **A variational-Bayes technique for aggregating probabilistic PCA mixtures from their parameters**. Rapport de recherche, version étendue de ICPR'2010, 30 pages, soumis à une revue, http://hal.archives-ouvertes.fr/docs/00/47/60/76/pdf/pami.pdf, HAL, Juin 2010b
- P. Bruneau. **De l'utilisation pratique des mélanges de gaussiennes**. *Journée des doctorants de l'école doctorale STIM (JDOC'09) - prix du meilleur article (sur env. 100)*, Nantes (France), Avril 2009

1.4 Graphe des publications

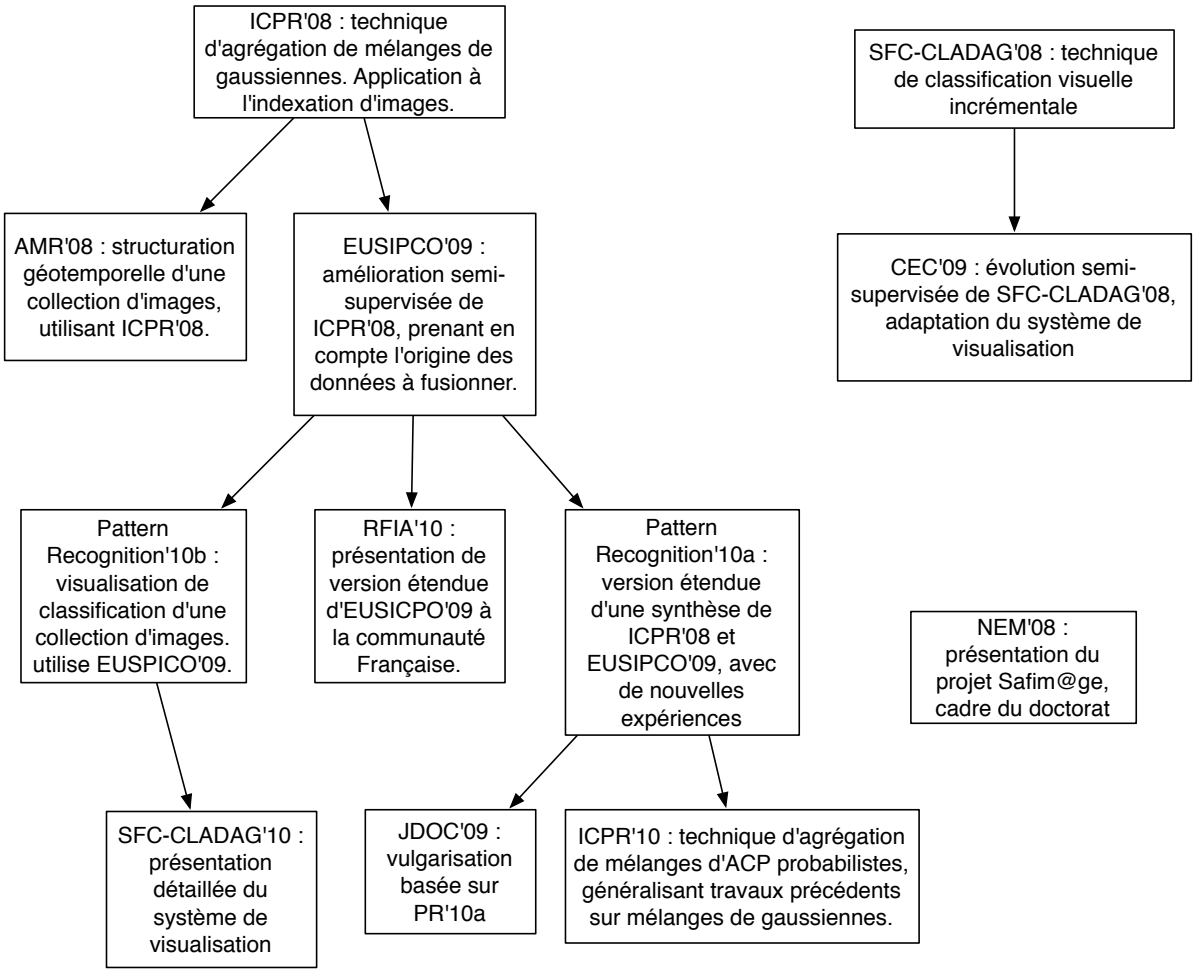


FIG. 1.1 – Graphe présentant nos publications et les liens entre celles-ci.

1.5 Organisation du document

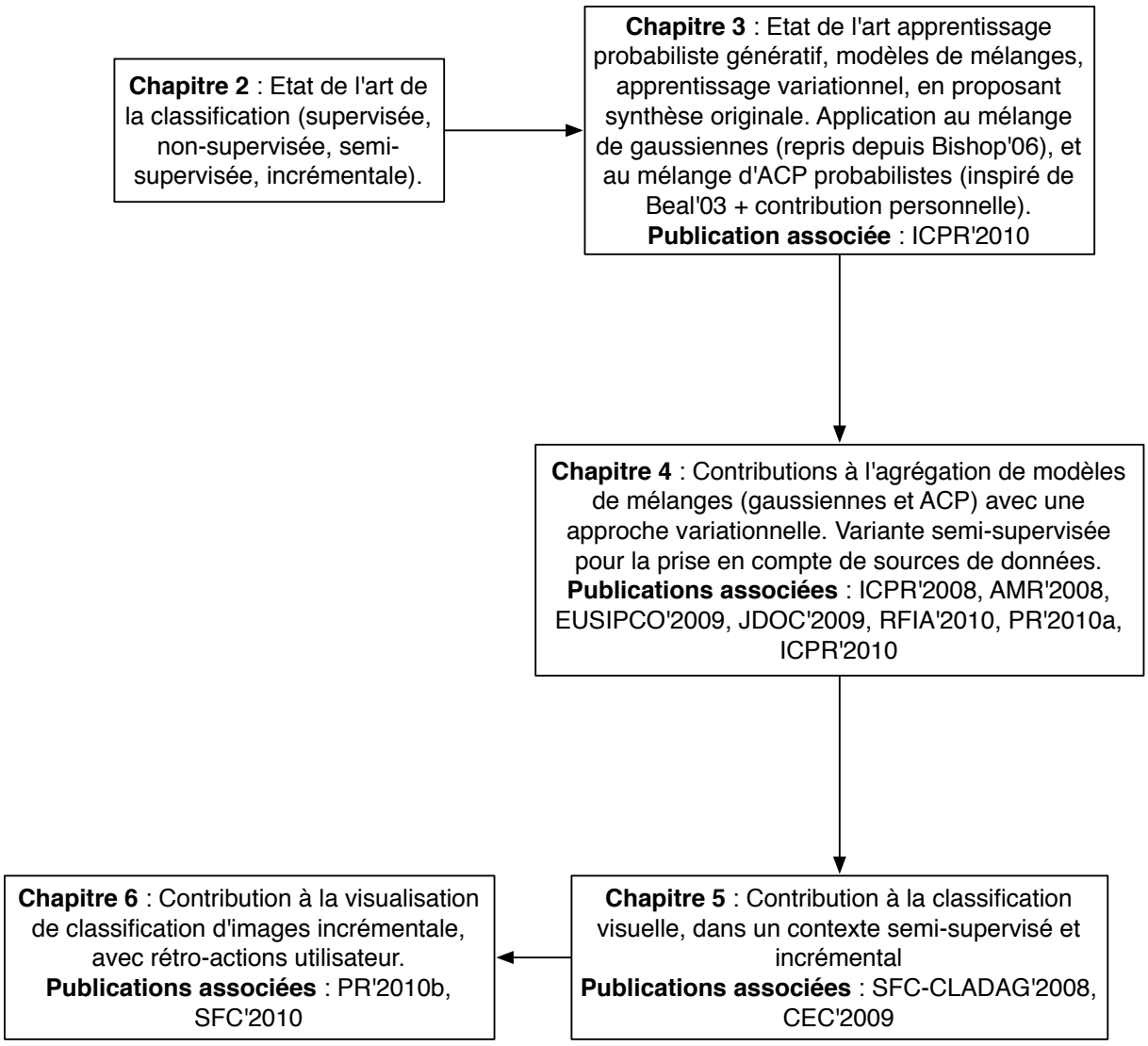


FIG. 1.2 – Graphe présentant l'organisation du document.

Chapitre 2

Classification automatique : un état de l'art

2.1 Introduction

Dans ce chapitre, nous présentons un état de l'art introductif des techniques de classification. Ce faisant, nous entendons préciser le cadre de nos contributions. Cela nous permettra aussi d'envisager les évolutions des techniques de classification, d'un cadre classique vers l'adaptation aux évolutions technologiques les plus récentes. Des techniques tirant notamment parti de recommandations d'utilisateurs, ou de vastes collections de données en perpétuelle évolution, éventuellement distribuées sur un réseau, sont présentées.

Dans un premier temps, nous évoquons les représentations possibles pour les éléments de données. Nous précisons alors le cadre restrictif que nous avons choisi pour notre document. Nous proposons ensuite une description classique du problème de classification. En particulier, les approches supervisée et non-supervisée sont brièvement évoquées. Cette présentation nous permet notamment d'illustrer la variété des modèles possibles, avec notamment les modèles de classification hiérarchique. Nous poursuivons avec la description de l'approche semi-supervisée, ayant émergé récemment en tant que solution intermédiaire aux points de vue classiques. Les méthodes incrémentales sont ensuite brièvement présentées. Enfin, nous évoquons les possibilités d'adaptation de l'apprentissage au cas où les données sont distribuées, et non centralisées comme cela est en général supposé dans l'approche classique.

2.2 Représentation d'une collection de données

2.2.1 Éléments

Avant d'appliquer une méthode de classification sur une collection d'éléments de données, il faut choisir un espace de représentation pour ceux-ci. Le modèle le plus simple, que nous utiliserons le plus souvent dans ce document, est de représenter une population d'individus comme des valeurs dans $\mathbb{R}^d$:

Identifiant	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
A	5.1	3.8	1.6	0.2
B	5.1	3.5	1.4	0.3
C	5.6	2.9	3.6	1.3
D	4.5	2.3	1.3	0.3
E	6.7	3.3	5.7	2.5
F	6.7	3.1	4.4	1.4
G	6.2	2.2	4.5	1.5
...				

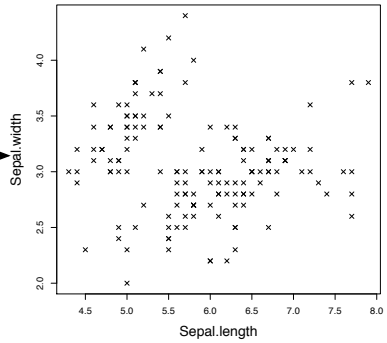


FIG. 2.1 – Échantillon d'individus de la base *iris*, définis sur $\mathbb{R}^4$, et représentation d'une projection sur les 2 premières dimensions.

Autrement dit, chaque individu est défini par d variables, chacune de celles-ci prenant ses valeurs dans $\mathbb{R}$. On peut aussi envisager le cas de d variables prenant leurs valeurs dans un ensemble fini de symboles :

Identifiant	Coût achat	Coût maintenance	Portes	Passagers	Coffre	Sécurité
A	high	med	3	4	big	high
B	med	med	3	2	med	high
C	vhigh	med	2	more	med	low
D	low	med	5more	2	big	low
E	vhigh	med	5more	4	small	high
...						

FIG. 2.2 – Échantillon d'individus de la base *car evaluation*.

Dans l'exemple ci-dessus, nous pouvons remarquer qu'une relation d'ordre peut être associée aux modalités de chaque variable. Ce n'est toutefois pas toujours le cas (e.g. couleur d'une voiture).

Notons qu'il peut parfois exister des ponts entre certaines de ces représentations. Considérons par exemple une collection de pages web. Pour des tâches de classification ou d'extraction de connaissances, le modèle *sac de mots* (ou *bag of words* dans la littérature) est communément employé. Celui-ci consiste à compter les occurrences

de mots d'un corpus dans chaque élément de la collection (i.e. chaque page web selon l'exemple cité). On passe alors d'une représentation symbolique à taille variable (i.e. séquence de mots) à une représentation avec un nombre de variables fixe, prenant leurs valeurs dans un ensemble dénombrable. De même, beaucoup d'algorithmes utilisent les fréquences d'apparition des mots [Nigam et al., 2000, Blei et al., 2003]. In fine, un glissement entre une représentation symbolique et $[0,1]^d$ est réalisé.

Au-delà des représentations élémentaires que nous venons d'évoquer, il est possible d'envisager des structures de représentation plus évoluées. Ainsi, les dépendances entre mots consécutifs dans un texte peuvent être considérées au lieu de l'hypothèse *sac de mots* (i.e. indépendance totale). Les objets peuvent être représentés dans le cadre de structures hiérarchiques (e.g. taxonomies). Chaque objet peut aussi être représenté par une fonction (e.g. un graphe [Bunke and Shearer, 1998], une fonction de l'espace L_2, ou une densité de probabilité, voir chapitre 6).

2.2.2 Similarités

La plupart des algorithmes repose sur des notions géométriques élémentaires, comme des calculs de distances, de similarités (i.e. pas nécessairement métriques), ou encore de centroïdes (i.e. barycentres). Quand les éléments sont représentés sur $\mathbb{R}^d$, la distance Euclidienne est le choix le plus immédiat. Assez fréquente dans le contexte du traitement de contenus textuels, la distance du cosinus est basée sur le produit scalaire de nos vecteurs de $\mathbb{R}^d$ (restreint à $[0,1]^d$ dans le cas de fréquences d'apparition de mots).

Certaines méthodes ne nécessitent pas d'hypothèse explicite sur la représentation des objets, mais seulement l'existence d'une similarité $\in [0,1]$ pour tout couple d'objets. Les méthodes spectrales, les méthodes à noyaux (e.g. SVM, voir section 2.3.4), ou encore certaines méthodes de classification visuelle (e.g. FClust [Picarougne et al., 2007], voir chapitre 5) rentrent dans cette catégorie.

Les similarités (ou de manière équivalente, dissimilarités) entre représentations complexes sont le plus souvent évaluées grâce à des fonctions spécialement adaptées. Par exemple, les mesures de divergence sont communément employées entre objets de la famille des distributions de probabilité. Les mesures de similarité entre objets représentés par des graphes ont également été étudiées [Bunke and Shearer, 1998].

Les représentations utilisées sont parfois difficiles à exploiter pour la visualisation. Dans ce cas, il existe des techniques permettant la construction d'une représentation $\mathbb{R}^d$ à partir des seules similarités, autorisant ainsi une visualisation facile avec des plans de projection. Celles-ci peuvent être basées sur une adaptation de MDS (*Multi Dimensional Scaling*) [Tenenbaum et al., 2000], ou sur la découverte d'une topologie en exploitant les voisinages des objets [Roweis and Saul, 2000].

Dans ce document, en l'absence de précision explicite, nous supposerons l'existence d'une représentation des objets sur $\mathbb{R}^d$.

2.3 Approche classique à la classification

2.3.1 Modèle

Soit un ensemble d'individus $\mathbf{X} = \{\mathbf{x}_n\}$, $n \in 1, \ldots, N$, que par simplicité nous supposerons définis sur $\mathbb{R}^d$. Une fonction de classification (ou parfois *classifieur*, dans la littérature) associe, à chaque individu, une étiquette (ou classe) dans un ensemble fini de symboles $E = \{c_1, \ldots, c_K\}$:

$$f : \mathbb{R}^d \to E \tag{2.1}$$
$$\mathbf{x}_n \to f(\mathbf{x}_n) \tag{2.2}$$

Considérons de nouveau l'ensemble d'individus $\mathbf{X}$, et associons-y une variable aléatoire représentant l'ensemble des étiquettes respectives, $\mathbf{Y} = \{y_n\}$. Deux applications différentes de ce modèle général ont été classiquement considérées dans la littérature :

- **l'approche supervisée** : $\mathbf{X}$ et $\mathbf{Y}$ sont connus et fournis au système. L'objectif est alors de trouver f respectant au mieux cette connaissance.
- **l'approche non-supervisée** : $\mathbf{X}$ est connu et $\mathbf{Y}$ inconnu. L'objectif est de déterminer à la fois f et $\mathbf{Y}$.

L'ensemble $\mathbf{X}$ (ainsi que $\mathbf{Y}$, si celui-ci est connu) utilisé pour la construction de f est appelé l'*échantillon d'apprentissage*.

2.3.2 Approche supervisée

Idéalement f est telle que $y_n = f(\mathbf{x}_n)$, $\forall n$. Toutefois, une telle fonction n'existe pas toujours ; autrement dit, l'échantillon n'est pas toujours séparable par f selon ses étiquettes. En particulier, si f est une fonction linéaire de $\mathbf{x}$, nous dirons alors que l'échantillon n'est pas *linéairement séparable*. Ce qui signifie qu'en général, on souhaite une fonction f minimisant une erreur :

$$f_{\text{apprise}} = \arg\min_f \sum_n^N \text{erreur}(f(\mathbf{x}_n), y_n)$$

Le plus souvent une fonction d'erreur quadratique est utilisée dans la littérature ; la fonction f est alors déterminée (ou apprise) selon :

$$f_{\text{apprise}} = \arg\min_f \frac{1}{2} \sum_n^N \big(f(\mathbf{x}_n) - y_n\big)^2 \tag{2.3}$$

Ce type de fonction est parfois appelé *discriminante*. Elle peut être utilisée sur de nouveaux individus (i.e. n'appartenant pas à l'échantillon d'apprentissage), dont on voudrait découvrir la classe.

Ce formalisme définit implicitement les notions de *frontières et régions de décision* ; chaque classe, ou étiquette, c_k sera associée à une réunion d'ouverts de $\mathbb{R}^d$ $\mathcal{A}_k$ telle que $\mathbf{x}_n \in \mathcal{A}_k \rightarrow f(\mathbf{x}_n) = c_k$. On considère en général des classes de fonctions f n'associant, de manière déterministe, qu'une et une seule classe possible à une valeur de $\mathbb{R}^d$ donnée ; les $\mathcal{A}_k$ sont donc disjoints. Les $\mathbf{x}_n$ pour lesquels $f(\mathbf{x}_n)$ ne peut pas être décidée sont à la frontière d'au moins deux de ces disjoints ; ils constituent les frontières de décision.

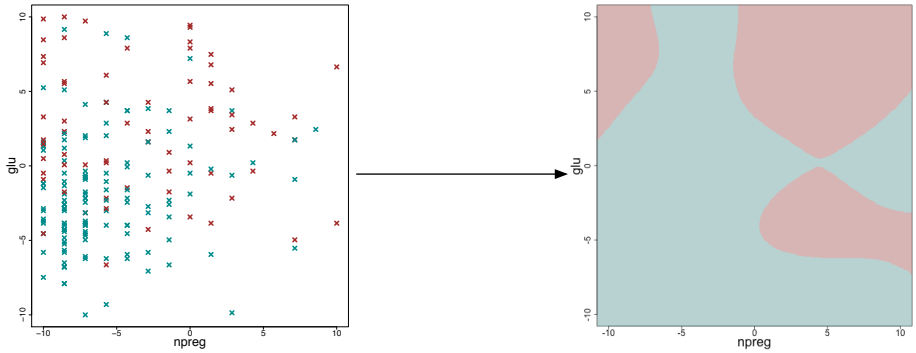


FIG. 2.3 – Échantillon *pima* [UCI, 2007], et régions de décision déterminées sur celui-ci.

Les méthodes de classification sont souvent distinguées suivant le procédé employé pour l'affectation d'un élément $\mathbf{x}_n$ à une région de décision :

- **Approches «à plat»** : l'affectation peut résulter de la réponse d'une fonction sophistiquée à l'élément qui lui est présenté (e.g. SVM [Vapnik, 1995, Burges, 1998], réseaux de neurones [Bishop, 2006, chap.5]).
- **Approches hiérarchiques** : l'affectation résulte d'une succession de décisions, orientant vers une portion spécifique de l'espace (e.g. arbres de décision [Breiman et al., 1984, Quinlan, 1986, 1993], mélanges d'experts [Jordan and Jacobs, 1994, Bishop and Svensén, 2003]).

Dans ce document, nous traiterons essentiellement d'approches non-hiérarchiques. Notons cependant que des modèles hiérarchiques peuvent être constitués en combinant des modèles non-hiérarchiques (e.g. mélanges d'experts).

2.3.3 Approche non-supervisée

La classification non-supervisée est parfois identifiée par le terme *clustering* dans la littérature. Nous préférerons parler d'approche non-supervisée dans ce document, sauf quand les dénominations imposées par la littérature ne nous laissent pas le choix. Rappelons que dans ce contexte, $\mathbf{Y}$ est inconnu. Ceci signifie que pour apprendre f, on ne peut pas exploiter une fonction d'erreur comme (2.3).

Beaucoup de méthodes sont basées sur la maximisation de l'inertie (i.e. variance) interclasse et la minimisation de l'inertie intraclasse. Autrement dit, les classes que détermine l'algorithme de classification doivent être «compactes» est bien séparées. En pratique, ces caractéristiques seront constitutives du critère optimisé par l'algorithme.

L'algorithme K-means [McQueen, 1967] est l'exemple le plus classique mettant en oeuvre ces principes d'inertie. Son fonctionnement est illustré dans la figure 2.4.

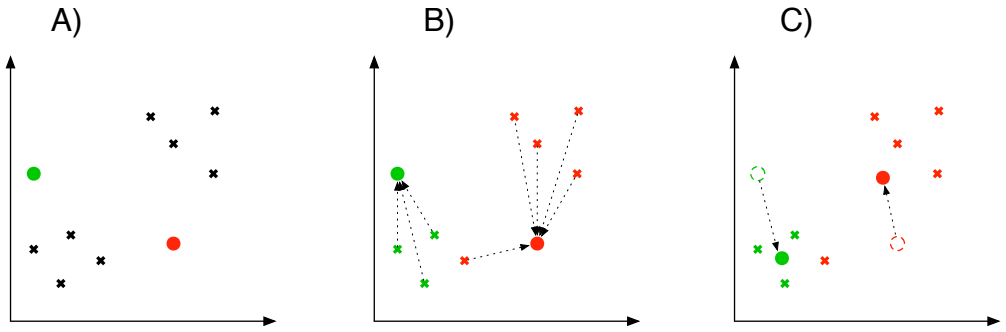


FIG. 2.4 – A) Échantillon 2D, et 2 centroïdes initialisés aléatoirement. B) chaque élément de l'échantillon est affecté au centroïde le plus proche. C) chaque centroïde est mis à jour avec la moyenne des éléments lui étant affectés.

Cet algorithme est un exemple de méthode non-hiérarchique dans un contexte non-supervisé. En effet, après convergence, une partition de la collection d'éléments (ainsi que, très souvent, de l'espace des données) est obtenue.

Des approches hiérarchiques comme la classification ascendante hiérarchique (CAH) [Ward, 1963, Murtagh, 1984] procèdent par agglomération des éléments selon un critère donné (e.g. minimiser la perte d'inertie interclasse). De ce processus résulte un dendrogramme (voir figure 2.5). Les éléments agglomérés sont indexés sur la ligne du bas, et la hauteur des arêtes regroupant deux éléments est proportionnelle au coût relatif selon le critère choisi. Des partitions imbriquées peuvent être obtenues à partir de ce graphe en «coupant» celui-ci à une hauteur donnée. Cette partition est fine ou grossière selon la hauteur choisie pour la coupe.

2.3.4 Validation des fonctions de classification

Classiquement, après avoir appris une fonction de classification sur un échantillon d'apprentissage, on estime la qualité de celle-ci sur un échantillon de test, ou de validation. Ceci permet de favoriser des fonctions avec une bonne capacité de généralisation.

Dans le cas d'une méthode de classification supervisée, l'erreur entre les étiquettes de l'échantillon de test et les prédictions est le plus souvent utilisée. Dans un contexte non-supervisé mettant en oeuvre un modèle probabiliste, la vraisemblance d'un échantillon

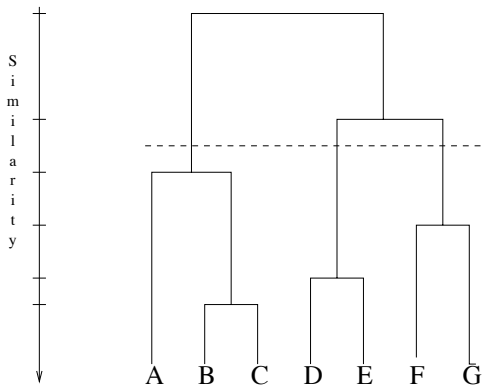


FIG. 2.5 – Exemple de dendrogramme. Un exemple de coupe est proposé en pointillés (extrait de [Jain et al., 1999]).

de test peut être employée pour évaluer la qualité du modèle. Ces principes sont illustrés sur les figures 2.6 et 2.7.

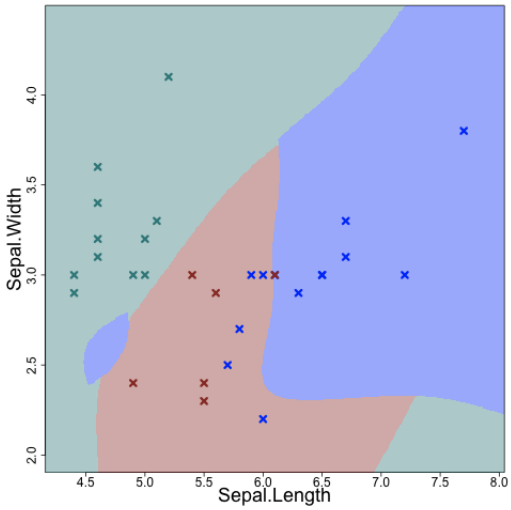


FIG. 2.6 – Comparaison entre ensemble de validation et régions de décision pour les données *iris* [UCI, 2007].

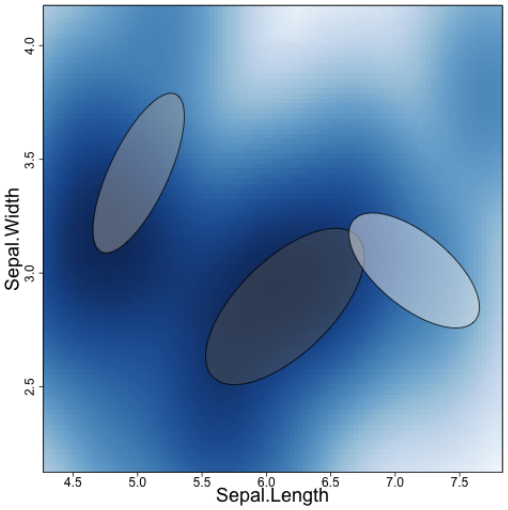


FIG. 2.7 – Densité d'échantillon de test et composantes d'un mélange de gaussiennes, pour validation selon vraisemblance.

De manière alternative, voire complémentaire, un ensemble d'étiquettes réelles peut être comparé aux affectations réalisées de manière non-supervisée avec un critère approprié. En effet, l'erreur de classification utilisée dans le contexte supervisé n'a alors pas de sens, étant donné qu'un apprentissage non-supervisé ne saurait avoir connaissance a priori du nombre et de la nature des classes. Outre un critère de pureté, il peut être

souhaitable de comparer les groupements effectués par la fonction de classification et les étiquettes réelles.

Par exemple, une mesure d'erreur pour la classification non-supervisée, proposée dans [Fowlkes and Mallows, 1983, Azzag, 2005, Picarougne et al., 2007] exploite les valeurs d'étiquettes réelles et calculées pour tous les couples possibles d'éléments de l'échantillon de validation.

Afin d'assurer une meilleure robustesse statistique à l'estimateur de qualité, le processus de validation est parfois découpé en plusieurs étapes. Plusieurs évaluations successives sont effectuées sur des sous-échantillons aléatoires de l'ensemble de test, et la qualité du classifieur est estimée suivant la combinaison des résultats (e.g. moyenne, indication de confiance, etc). C'est la validation croisée (voir par exemple [Kohavi, 1995]).

Maintenant supposons que la fonction f à apprendre est paramétrée par un vecteur $\mathbf{w}$ de taille variable. On voit alors clairement qu'une fonction avec un vecteur $\mathbf{w}$ de taille N sera capable de facilement apprendre parfaitement les étiquettes d'un ensemble d'apprentissage. Mais une telle fonction sera rarement efficace en généralisation. On appelle couramment ce phénomène le *sur-apprentissage* : le modèle appris est alors trop complexe relativement au problème à résoudre.

Considérons le cas de la classification supervisée basée sur les réseaux de neurones [Bishop, 2006, chap.5]. Pour ce type de fonctions de classification supervisée, le paramétrage est effectué par la couche cachée du réseau de neurones. Plus précisément, le vecteur $\mathbf{w}$ sera constitué des poids associés à chaque arc du réseau de neurones ; notons, sur la figure 2.8, que la taille de cet ensemble est directement liée au nombre de neurones de la couche cachée.

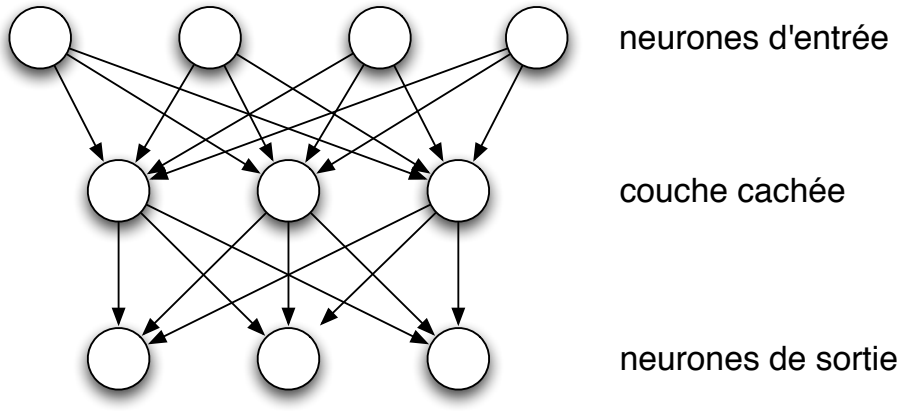


FIG. 2.8 – Exemple de réseau de neurones.

Sur la figure 2.9, nous présentons des réseaux de neurones déterminés sur un échantillon

d'apprentissage de Pima, pour des tailles de couche cachée variables. Nous voyons qu'un réseau de neurones avec une couche cachée importante (i.e. un modèle complexe) apprend parfaitement l'échantillon d'apprentissage, mais sera en général assez mauvais en généralisation. A contrario, un modèle de complexité limitée admet des erreurs à l'apprentissage, mais sera souvent plus robuste en validation.

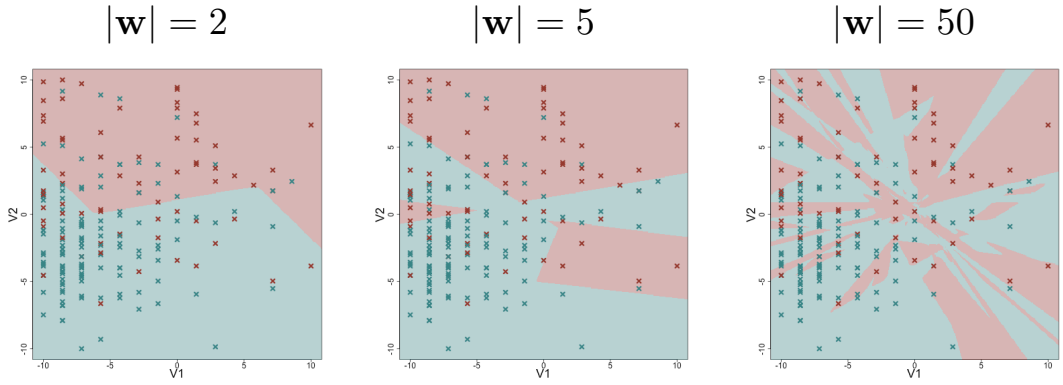


FIG. 2.9 – Réseaux de neurones de complexité variable sur l'échantillon d'apprentissage de *Pima*.

La manière la plus immédiate d'augmenter les chances d'avoir une validation satisfaisante est de *régulariser* la fonction de classification. Par exemple, dans le cas d'une méthode de classification supervisée basée sur la mesure d'erreur (2.3), une régularisation possible est :

$$\mathbf{w}_{\text{régularisé}} = \arg\min_{\mathbf{w}} \frac{1}{2} \sum_{n}^{N} \big(f(\mathbf{x}_n, \mathbf{w}) - y_n\big)^2 + \frac{\lambda}{2}||\mathbf{w}||^2 \tag{2.4}$$

Autrement dit, la magnitude du vecteur de paramètre est limitée autant que possible. La régularisation consiste donc à chercher un compromis entre une erreur faible et un modèle peu complexe. C'est le principe du rasoir d'Occam : l'ensemble de paramètres ne doit pas être plus complexe que ne l'impose la nature du problème traité.

L'erreur d'apprentissage est parfois définie comme le risque empirique. Le risque réel, ou risque attendu, est alors l'espérance de l'erreur de validation Guyon et al. [1992], Vapnik [1995]. Ce risque dépend de la dimension VC (Vapnik Chervonenkis), intimement liée au pouvoir modélisant (i.e. complexité, ou taille des paramètres) d'une famille de fonctions, même si quelques exceptions doivent être considérées Burges [1998]. Un bon modèle n'est alors pas celui qui minimise l'erreur sur un ensemble d'apprentissage, mais celui qui minimise l'erreur potentielle sur un ensemble de test.

L'approche bayésienne présente un point de vue quelque peu orthogonal par rapport à cette dernière définition. Celle-ci est présentée dans un cadre probabiliste : pour

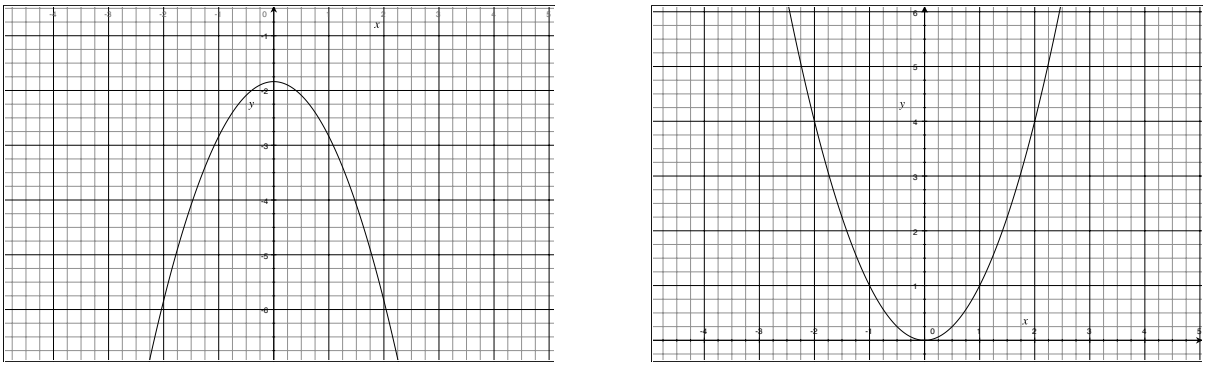


FIG. 2.10 – À gauche : log vraisemblance selon $\mathcal{N}(0,1)$. À droite : erreur quadratique par rapport à 0.

apprendre une fonction, au lieu de minimiser une erreur, on maximise une vraisemblance (i.e. le produit des probabilités des éléments de l'échantillon, pour une définition détaillée voir section 3.2). Notons, au passage, que ces critères peuvent être équivalents sous certaines conditions. Nous présentons un exemple simple dans la figure 2.10 : nous y voyons que le logarithme de la vraisemblance d'un point selon la gaussienne $\mathcal{N}(0,1)$ et l'erreur quadratique de celui-ci selon la même moyenne suivent la même courbe, au signe près.

Dans un cadre bayésien, données et paramètres du modèle à estimer sont conjointement associés à des distributions de probabilité. La distribution portant sur les paramètres peut être qualifiée d'*a priori* pour le modèle à estimer. Classiquement, on assure une forme de régularisation en utilisant un a priori associant une vraisemblance d'autant plus faible que le modèle est complexe. La conséquence de ce principe est illustrée sur la figure 2.11 : on voit alors qu'un modèle plus complexe, donc plus flexible, peut modéliser des échantillons de données plus variés. Un modèle contraint, associé à moins de paramètres, peut modéliser moins d'échantillons potentiels. Mais le cas échéant, en utilisant un a priori celui-ci sera préféré.

Le SVM (Séparateur à Vaste Marges, ou *Support Vector Machine*) est une technique développée dans l'optique d'une optimisation du risque évoqué précédemment Vapnik [1995], Burges [1998]. La réalisation de l'apprentissage d'une fonction discriminante y est posé comme un problème d'optimisation combinatoire pour le choix des meilleurs vecteurs de *support*, i.e. les points de l'échantillon d'apprentissage définissant la plus grande marge entre les classes. Une illustration de ce principe est proposée sur la figure 2.12.

La valeur de discriminante d'un élément est obtenue par combinaison linéaire d'évaluations selon une fonction noyau (*kernel*, homogène à un produit scalaire dans un espace préhilbertien), appliquée à l'élément et à chaque vecteur de support. Les fonctions noyaux sont éventuellement non-linéaires, autorisant des frontières complexes. Un la-

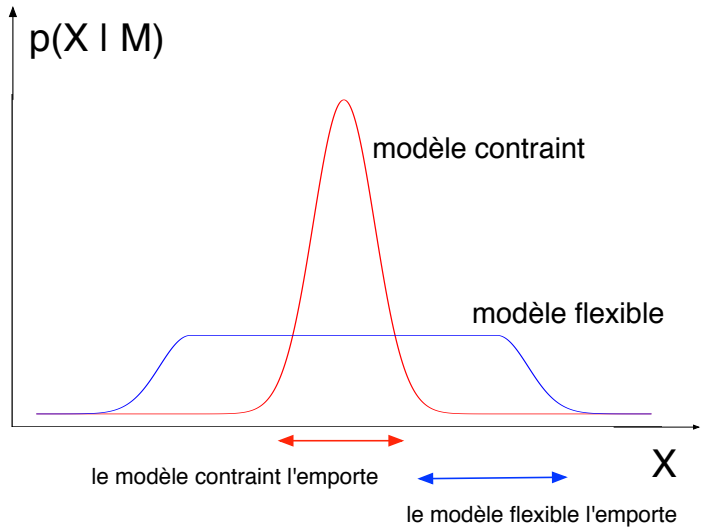


FIG. 2.11 – Représentation de la vraisemblance de modèles en fonction des échantillons possibles. L'utilisation d'a priori permet de réserver les modèles complexes aux situations où cela est nécessaire.

grangien est utilisé pour formaliser le problème à optimiser sous contrainte d'une marge maximale, et l'utilisation d'une famille de supports autorise une représentation duale du problème. On se retrouve alors avec un problème d'optimisation convexe. Pour des problèmes de taille réduite, la méthode de Newton ou autres techniques classiques de programmation non-linéaire peuvent être utilisées. Un état de l'art du cas général est proposé dans Burges [1998].

Un pendant probabiliste et bayésien du SVM a été proposé par Tipping Tipping [2001]. L'apprentissage des SVM souffre souvent du nombre trop élevé de vecteurs de support (déterminant de fait la taille du modèle) découverts par les algorithmes habituels : une adaptation bayésienne du problème a donc été formulée dans le but de limiter au mieux cet ensemble. La transformation du problème amène à la maximisation itérative d'une fonction de vraisemblance, proche dans l'esprit de l'algorithme EM Dempster et al. [1977].

2.4 Classification semi-supervisée

L'approche semi-supervisée à la classification consiste à supposer que l'échantillon utilisé pour l'apprentissage d'un modèle n'est que partiellement étiqueté. Définissons donc $\mathbf{X}_L = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\}$ (pour *Labelled*) la partie de l'ensemble d'apprentissage étiquetée par $\mathbf{Y}_L = \{y_1, \ldots, y_L\}$, et $\mathbf{X}_U = \{\mathbf{x}_{L+1}, \ldots, \mathbf{x}_{L+U}\}$ (pour *Unlabelled*) la partie sans

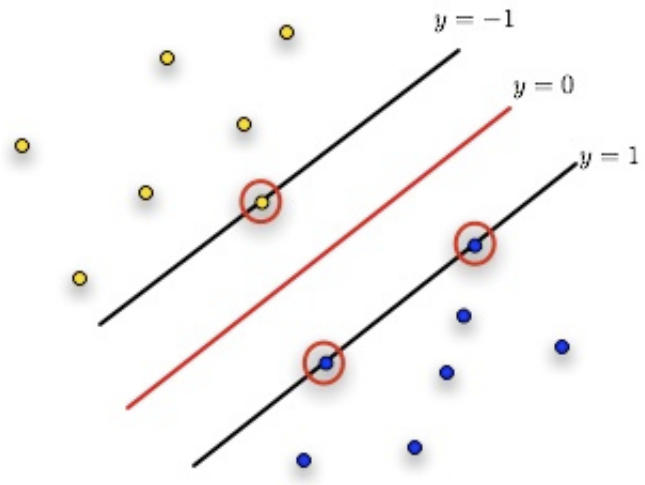


FIG. 2.12 – Représentation d'un SVM. Les vecteurs de support, entourés de rouge, définissent la marge maximale entre les deux classes d'individus (avec ici $l_1 = -1$ et $l_2 = +1$).

étiquette.

Le concept d'apprentissage semi-supervisé a émergé de l'approche supervisée à la classification. Il s'agissait alors d'utiliser la densité de $\mathbf{X}_U$ pour améliorer l'apprentissage d'une fonction discriminante.

Pour la plupart des applications du monde réel, il est en effet très difficile d'obtenir des données étiquetées (i.e. celles-ci ne peuvent être créées que manuellement) afin de former un ensemble d'apprentissage représentatif pour un entraîner une fonction discriminante, alors qu'obtenir des données non-étiquetées ne coûte quasiment rien. Aussi l'approche la plus immédiate est de chercher à améliorer une méthode de classification en utilisant l'information annexe fournie par les données non-étiquetées.

L'implémentation la plus populaire de cette approche est l'apprentissage transductif au moyen de SVM [Chapelle et al., 2006]. Les SVM traditionnels, entraînés avec un ensemble d'apprentissage trop petit mènent en général à de mauvaises performances en généralisation. L'apprentissage transductif utilise également la distribution globale sur les $\mathbf{x}_n$, afin de disposer d'un bon estimateur a priori des régions de l'espace des données à faible densité, guidant ainsi au mieux le positionnement des frontières de décision. Toutefois, l'optimisation de ce problème est difficile, car non-convexe [Chapelle et al., 2006], obligeant le recours à une approximation. Dans [Joachims, 2006], un SVM sous-optimal est d'abord entrainé en utilisant seulement $\mathbf{X}_L$, puis $\mathbf{X}_U$ est incorporé itérativement, cet ajout étant entrelacé de ré-apprentissages.

De manière alternative, on peut aussi voir la classification semi-supervisée comme le partitionnement de $\mathbf{X} = \mathbf{X}_L \cup \mathbf{X}_U$ avec des contraintes matérialisées par $\mathbf{Y}_L$, ou explicitement entre éléments considérés 2 à 2. Les méthodes de classification non-supervisée font souvent l'hypothèse que des individus faisant partie du même groupe sont susceptibles d'appartenir à la même classe ; c'est un autre point de vue sur l'hypothèse de *cluster*. Dans un contexte de classification, les modalités de classes présentes dans $\mathbf{X}_L$ peuvent ne pas couvrir toutes les valeurs possibles, ce qui signifie qu'une méthode de classification semi-supervisée est susceptible de découvrir des groupes d'étiquette encore inconnue. Ces observations sont résumées sur la figure 2.13.

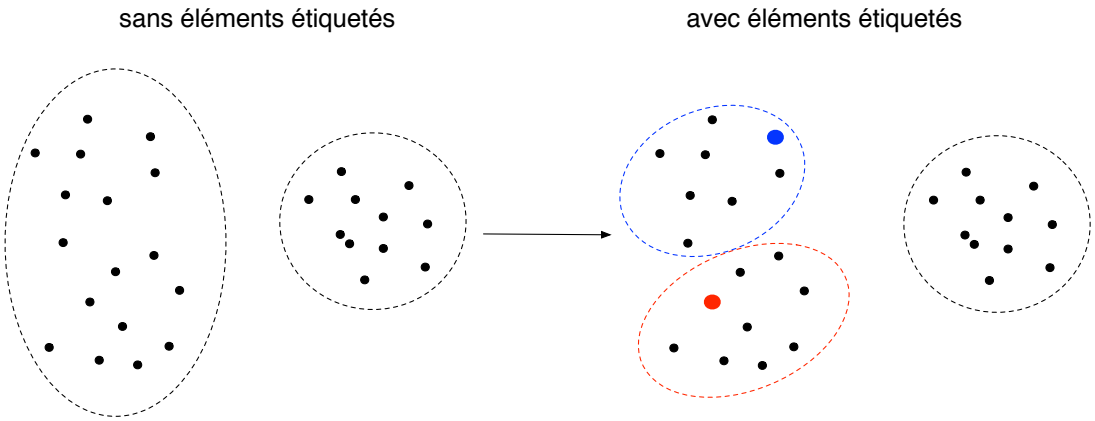


FIG. 2.13 – Exemple de mise en oeuvre d'une approche semi-supervisée pour la classification. Sans éléments étiquetés, 2 groupes semblent être adéquats. Avec l'apport d'un étiquetage partiel, les 3 groupes indiqués semblent mieux adaptés.

$\{\mathbf{X}_L, \mathbf{Y}_L\}$ définit donc implicitement un ensemble de contraintes, car souvent dans la littérature il est implicitement supposé que $y_i \neq y_j \rightarrow \text{groupe}(\mathbf{x}_i) \neq \text{groupe}(\mathbf{x}_j)$, pour tout couple d'éléments dans $\mathbf{X}_L$.

L'approche *cluster-and-label* est la plus immédiate [Chapelle et al., 2006] : un algorithme de partitionnement est appliqué sur $\mathbf{X}$, puis $\mathbf{X}_L$ et $\mathbf{Y}_L$ sont utilisés pour attribuer une modalité de classe chaque groupe. Plus généralement, un compromis doit être trouvé entre l'hypothèse de *cluster* et le respect des contraintes.

Miller et Uyar [Miller and Uyar, 1997] utilisent l'influence d'un ensemble de données étiquetées pour estimer un mélange de gaussiennes sur tout l'échantillon disponible. Ils introduisent des variables cachées distinguant la notion de classe et de composante, permettant d'estimer des classes portant sur plusieurs composantes gaussiennes (i.e. groupes). Un algorithme EM est utilisé afin d'obtenir un optimum local.

Law et al. [Law et al., 2004] estiment également un mélange de gaussiennes avec EM, mais leur modèle utilise des contraintes entre éléments plutôt que des étiquettes de classe explicites. Ils proposent une fonction de vraisemblance modifiée afin de prendre en compte le respect des contraintes. Cette fonction est déclinée depuis un DAG (Directed Acyclic Graph, (voir section 3.4.5), donc les méthodes d'inférence pour les réseaux bayésiens peuvent également être appliquées, comme par exemple la propagation de pertinence (*belief propagation*) ou la jonction d'arbres (*junction tree*).

Dans [Law et al., 2005], leur modèle est modifié afin de pouvoir paramétrer le compromis entre vraisemblance et respect des contraintes. La fonction objectif ainsi obtenue est optimisée en utilisant un algorithme variationnel local (l'approche variationnelle globale, sensiblement différente, est présentée section 3.5.1). Basu et al. [Basu et al., 2006] incorporent des contraintes entre éléments sous la forme de champs de Markov cachés sur les variables d'affectation. Ils définissent une fonction de vraisemblance présentant un compromis entre adaptation aux données et respect des contraintes. Plusieurs métriques sont possibles (Euclidienne, divergence KL, distance du cosinus). Cette fonction est optimisée via une variante d'EM. Les affectations (étape E) ne peuvent pas être effectuées via une expression analytique, aussi une optimisation par ICM (Iterated Conditional Modes) est utilisée.

Nigam et al. [Nigam et al., 2006] utilisent des données étiquetées afin de construire un modèle de classification initial. Leurs travaux sont appliqués à la classification de texte avec un mélange de distributions multinomiales, mais on peut facilement envisager un mélange de gaussiennes à la place afin d'étendre le champ d'application. Le partitionnement est ensuite réalisé de manière classique, excepté que les affectations des points étiquetés restent fixes. Des extensions pour gérer plusieurs groupes par classe, et pour éviter de mauvais optima locaux, sont également proposées.

2.5 Classification incrémentale

Les approches classiques à la classification supposent un échantillon de données disponible entièrement pour toute la durée de l'apprentissage, autorisant ainsi divers types d'itérations. Toutefois, dans certains dispositifs expérimentaux (e.g. réseaux de capteurs), les éléments de données ne sont présentés au système qu'une seule fois. Une méthode de classification incrémentale est adaptée à ce contexte particulier ; au lieu d'être déterminée selon les principes classiques d'apprentissage et de validation, la fonction de classification doit s'adapter en flux aux données reçues (voir figure 2.14).

Ce problème a été identifié très tôt dans la littérature, notamment avec COBWEB, une méthode de *clustering* conceptuel [Fisher, 1987]. Plus précisément, celle-ci procède à la construction incrémentale d'un arbre de concepts à partir d'éléments définis par des variables catégorielles, en ne mémorisant aucun des éléments présentés depuis le flux autrement qu'en les incorporant à la fonction de classification.

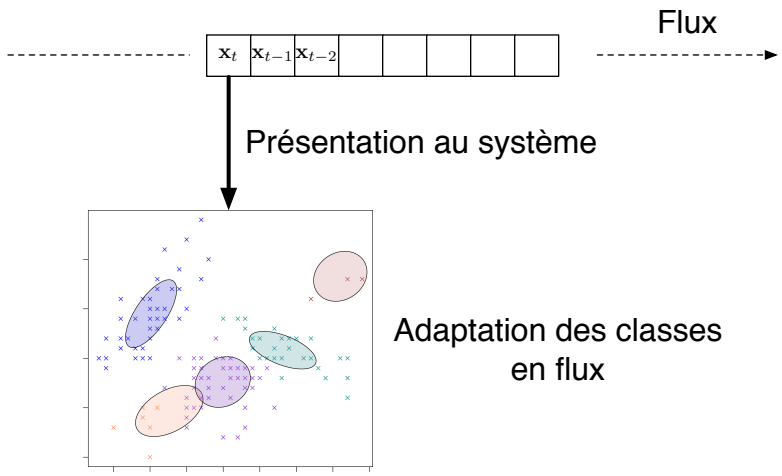


FIG. 2.14 – Présentation des données en flux.

Certains algorithmes de classification sont partiellement incrémentaux, comme BIRCH par exemple [Zhang et al., 1996]. En effet, cette méthode repose sur la définition et la construction incrémentale d'un résumé des données présentées au système (*cluster features*). Celui-ci est alors utilisé pour la construction d'une fonction de classification, éventuellement de manière non-incrémentale.

De nombreuses méthodes mettant originellement en oeuvre une approche classique, ont été adaptées au contexte de données en flux. Ainsi, des variantes incrémentales ont été proposées pour les algorithmes DBSCAN [Ester et al., 1998], K-means [Bottou and Bengio, 1995], et EM [Neal and Hinton, 1999].

Dans toutes ces approches se pose le problème d'une procédure «d'oubli». Celui-ci est résolu par diverses heuristiques, soit en contraignant la mémoire utilisée par le processus, soit en programmant l'affaiblissement de l'influence des éléments de données en fonction de leur horizon temporel. Ainsi, les *cluster features* de BIRCH réalisent un compromis entre la conservation de l'information et la place occupée par la structure. Plus simplement, une fenêtre glissante (voir figure 2.15), par exemple employée dans [Babcock et al., 2003, Lavergne et al., 2007], ne met à jour la fonction de classification en n'utilisant qu'un horizon temporel fini. Cet horizon fini est simulé de manière continue avec un paramètre «d'affaiblissement» ajouté aux algorithmes K-means [Bottou and Bengio, 1995] et EM [Neal and Hinton, 1999].

Ces procédures d'oubli ont également été formalisées dans un cadre probabiliste. Ainsi, chaque itération de mise à jour d'un filtre de Kalman [Kalman, 1960] traite l'élément $\mathbf{x}_t$ uniquement, puis l'oublie immédiatement. La mémorisation est alors effectuée au moyen de statistiques synthétiques d'un modèle probabiliste. Cette approche est généralisée au moyen d'un régresseur de taille fixe (au lieu d'un seul point) avec les modèles Auto-Régressifs (AR). Cette question sera quelque peu approfondie dans la

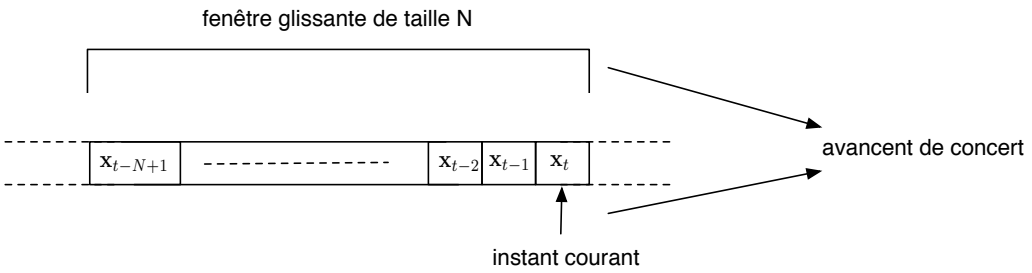


FIG. 2.15 – Illustration du principe de fenêtre glissante.

section 3.5.5.

2.6 Apprentissage sur données distribuées

Les techniques usuelles d'apprentissage de modèles de classification fonctionnent de manière centralisée : autrement dit, elles ne sont donc pas applicables directement au cas où les données sont réparties sur plusieurs sites. Nous commençons par noter qu'il est possible de répartir les données de deux manières «orthogonales» :

- **Verticalement** : les mêmes éléments sont décrits sur chaque site avec des attributs différents,
- **Horizontalement** : des éléments différents sont stockés sur chaque site. En général ils sont décrits par les mêmes attributs, mais cette contrainte est levée avec certaines méthodes.

Nous illustrons ces deux possibilités simplement dans la figure 2.16.

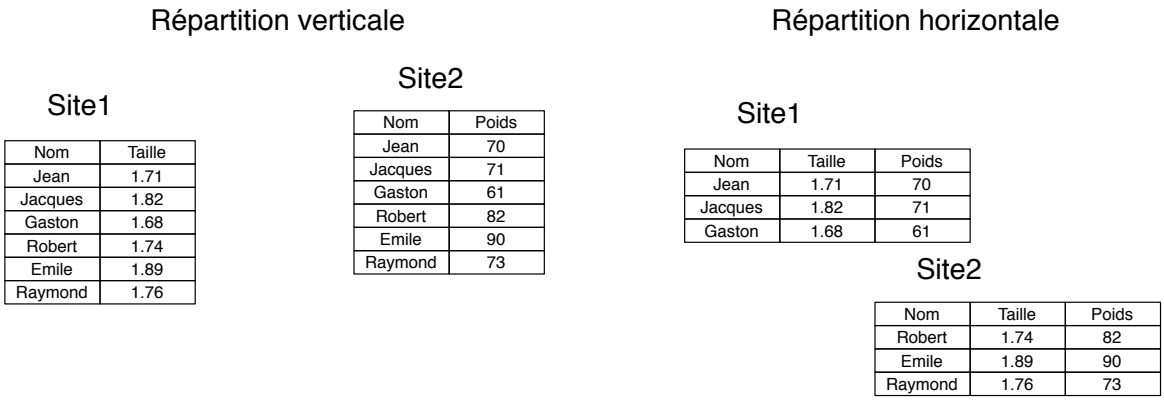


FIG. 2.16 – Exemples de répartition d'éléments de manière verticale ou horizontale.

Pour traiter le cas de la répartition verticale, les méthodes de recherche de consensus dans les partitions (*consensus clustering*) sont en général employées. Ce problème est NP-difficile [Gionis et al., 2007], ce qui motive l'utilisation de méthodes approchées. Dans [Strehl and Ghosh, 2003], cette situation est approchée par un problème d'optimisation combinatoire plus simple, où un critère d'information partagée est utilisé. Ce même problème peut également être considéré selon des mesures de distance entre partitions [Gionis et al., 2007]. Plusieurs heuristiques pour la résolution, ainsi que leurs propriétés théoriques, sont proposées. Dans [Ailon et al., 2008], le consensus des partitions est présenté comme une spécialisation du problème de l'agrégation de classements dans un tournoi. Une représentation par un graphe orienté est utilisée, et un algorithme en temps polynomial optimisant un coût approché est proposé.

Remarquons que ces dernières approches ne sont pas spécifiques aux données réparties verticalement : on peut aussi très bien réaliser le consensus de partitions réalisées par plusieurs méthodes différentes sur les mêmes individus, définis sur les mêmes attributs. Certaines approches ont été adaptées aux cas des données réparties horizontalement en autorisant des étiquettes inconnues [Strehl and Ghosh, 2003] : mais un recouvrement explicitement défini entre des éléments de sites différents est alors nécessaire.

Dans cette thèse, nous considèrerons plutôt des données réparties horizontalement, et pour lesquelles aucune mise en correspondance d'éléments de sites distincts sera réalisée. Cela peut être pour des raisons de respect d'anonymat [Merugu and Ghosh, 2003].

Mais l'effet recherché peut également être la parcimonie dans les coûts de transmission et de calcul : quand les données stockées sur chaque site sont très volumineuses, il est crucial de ne transmettre que des représentations synthétiques dont la taille ne dépend pas de celle des échantillons traités. L'objectif est alors d'être capable d'agréger les modèles de classification (par exemple un ensemble de prototypes de groupes, ou un mélange de gaussiennes) en respectant au mieux l'information transmise par chacun des sites. Notons que cette information, et la perte éventuellement engendrée, ne peut être mesurée que vis-à-vis des modèles traités : le serveur central réalisant l'agrégation n'aura en effet pas accès aux données ou partitions originales, seulement aux modèles reçus.

A cette fin, dans [Januzaj et al., 2003], les auteurs proposent une variante de l'algorithme DBSCAN [Ester et al., 1996] ne nécessitant la transmission que de quelques prototypes représentatifs de chaque site vers un serveur central. Dans [Bechchi et al., 2007], une méthode de fusion de résumés hiérarchiques de bases de données est proposée. Ces résumés sont construits grâce aux méthodes de *clustering* conceptuel.

Dans cette section, nous traitons essentiellement de la construction de consensus entre plusieurs partitions. Cette approche est à contraster avec la comparaison de partitions, présentée notamment dans [Cabanes and Bennani, 2009], dont la mise en oeuvre permettrait plutôt de détecter des changement profonds dans la nature des données traitées, ou encore de déterminer quels sites sont similaires.

L'agrégation de modèles de classification est un sujet classique, mais les applications récentes en renouvellent l'intérêt ; notamment, quand les tâches d'apprentissage automatique et de reconnaissance de formes sont transposées sur des architectures distribuées (grilles de calcul, pair-à-pair).

Parmi les applications associées, on peut trouver le passage à l'échelle de moteurs de recherche parmi des contenus multimédia structurés [Ponce et al., 2006], ou encore l'estimation d'un modèle depuis un réseau de capteurs [Nowak, 2003]. Des structures de données (e.g. arbres) manipulant des modèles probabilistes peuvent aussi nécessiter la fusion de ceux-ci [Vasconcelos, 2001].

Dans ce document, nous traiterons en particulier de modèles génératifs, et de la manière de réaliser un apprentissage distribués en les utilisant. Un état de l'art spécifique à cette classe de modèles pour le cas distribué est donc proposé dans la section 4.2.

2.7 Conclusion

Dans cet aperçu de la typologie de la classification automatique, nous avons bien distingué les points de vue adoptés par les approches supervisée et non-supervisée. Nous avons ensuite présenté une approche intermédiaire et émergente, où l'on tente d'exploiter conjointement une information fournie par un utilisateur, et la distribution générale des données traitées. Nous avons également évoqué des méthodes de classification incrémentale. De même, après avoir précisé comment envisager des données distribuées, nous avons entrevu quelques techniques récentes conçues pour tirer parti de ce type d'architecture.

Chapitre 3

Modèles de mélange et approche variationnelle

3.1 Introduction

Dans ce chapitre, nous nous concentrons sur la vision génératrice de l'apprentissage, i.e. l'hypothèse que les données traitées sont associées à un modèle probabiliste. Après avoir présenté le vocabulaire et les concepts spécifiques à cette approche, nous introduisons les modèles de mélange, et l'utilisation qui peut en être faite, en particulier pour réaliser la classification d'un échantillon de données. Nous traitons ensuite de l'estimation de ceux-ci, et des problèmes alors soulevés. Nous constatons qu'une approche bayésienne est nécessaire dans ce contexte, et présentons le principe variationnel comme moyen de la mise en oeuvre de cette approche.

Tout d'abord, nous présentons donc l'approche génératrice d'un point de vue général. Nous introduisons ensuite les modèles de mélange, et les principes de leur estimation. En particulier, l'algorithme EM pour le mélange de gaussiennes est détaillé. Après avoir évoqué les limites de la version élémentaire de cet algorithme, nous détaillons le principe variationnel, amenant alors à un algorithme EM modifié présentant de meilleures propriétés. Les applications au mélange de gaussiennes (reprise depuis [Bishop, 2006, chap. 10]), et au mélange d'ACP probabilistes (inspirée de [Beal, 2003], augmentée de contributions personnelles [Bruneau et al., 2010c,b]) sont enfin données à titre d'illustration.

3.2 Approche génératrice

3.2.1 Définition

Soit $\mathbf{X} = \{\mathbf{x}_n | n \in 1, \ldots, N\}$, un échantillon de vecteurs à valeurs dans $\mathbb{R}^d$. L'approche génératrice à l'apprentissage automatique consiste à définir une densité de probabilité p sur $\mathbb{R}^d$, et supposer que $\mathbf{X}$ est issu de N tirages indépendants de cette loi de probabilité. Autrement dit, on modélise un échantillon de données par une loi de probabilité. On définit la *vraisemblance* de l'échantillon $\mathbf{X}$ selon la loi p :

$$p(\mathbf{X}) = \prod_{n=1}^{N} p(\mathbf{x}_n) \tag{3.1}$$

Ainsi que la *log-vraisemblance* associée :

$$\mathcal{L}(\mathbf{X}) = \ln p(\mathbf{X}) = \sum_{n=1}^{N} \ln p(\mathbf{x}_n) \tag{3.2}$$

3.2.2 Domaine de l'échantillon

On notera que le domaine de $\mathbf{x}$ peut être restreint à une partie de $\mathbb{R}^d$, sous condition d'existence d'une densité de probabilité adéquate. De manière générale, n'importe quel espace probabilisé fera l'affaire. On remarquera aussi qu'une simple transformation permet de traiter des données discrètes modales dans le cadre d'une approche génératrice ; chaque modalité de variable discrète sera associée à une dimension, et chaque valeur possible sera associée à un vecteur canonique dans $\mathbb{R}^d$ ([Blei et al., 2003], voir aussi définition (3.16)).

3.2.3 Famille exponentielle

Cette section est essentiellement inspirée de Bishop [Bishop, 2006, chap. 2].

Une distribution est dite de la famille exponentielle si elle est de la forme :

$$p(\mathbf{x}|\theta) = h(\mathbf{x})g(\theta)\exp\{\theta^T \mathbf{u}(\mathbf{x})\} \tag{3.3}$$

où l'on a défini le vecteur de paramètres θ de p. Cette famille de distributions a plusieurs propriétés intéressantes :

- La forme fonctionnelle de p est définie par la taille de θ,
- La fonction g peut être interprétée comme une constante de normalisation ; en effet, comme (3.3) est une densité de probabilité, on a :

$$g(\theta) \int h(\mathbf{x})\exp\{\theta^T \mathbf{u}(\mathbf{x})\} d\mathbf{x} = 1 \tag{3.4}$$

D'autre part, pour toute distribution de la famille exponentielle, il existe une distribution conjuguée, définie sur le domaine de θ, par :

$$p(\theta|\chi, \nu) = f(\chi, \nu)g(\theta)^{\nu}\exp\{\nu\theta^T \chi\} \tag{3.5}$$

(3.5) peut être interprétée comme une distribution *a priori* sur θ. En l'absence d'un échantillon de données, c'est en effet la seule information que l'on a sur les valeurs probables pour nos paramètres θ.

Maintenant, en considérant un échantillon $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ distribué selon (3.3) de manière *indépendante et identique* (i.i.d) , il suffit d'appliquer la règle de Bayes pour obtenir la distribution *a posteriori* sur θ :

$$p(\theta|\mathbf{X}, \chi, \nu) = \frac{p(\theta|\chi, \nu)p(\mathbf{X}|\theta)}{p(\mathbf{X})} \propto p(\theta|\chi, \nu)p(\mathbf{X}|\theta) \tag{3.6}$$

$$p(\theta|\mathbf{X}, \chi, \nu) \propto g(\theta)^{\nu+N} \exp\Big\{\theta^T\big(\nu\chi + \sum_n \mathbf{u}(\mathbf{x}_n)\big)\Big\} \tag{3.7}$$

avec :

$$p(\mathbf{X}|\theta) = \prod_n p(\mathbf{x}_n|\theta) \tag{3.8}$$

$$p(\mathbf{X}, \theta) = p(\theta) \prod_n p(\mathbf{x}_n|\theta) \tag{3.9}$$

$$p(\mathbf{X}) = \int p(\mathbf{X}, \theta) d\theta \tag{3.10}$$

Les formules (3.6)-(3.10) seront fondamentales pour la mise en oeuvre de l'apprentissage bayésien [Kass and Raftery, 1993]. Attention toutefois à la sémantique des notations : seule (3.8) est homogène avec (3.1). (3.10) est une constante obtenue par intégration selon toute une famille de distributions. Cette quantité sera souvent nommée *vraisemblance marginale*.

Ces autres faits seront importants par la suite :

- On voit que la distribution a posteriori (3.7) ne dépend des données $\mathbf{X}$ qu'au travers de *statistiques synthétiques* $\sum_n \mathbf{u}(\mathbf{x}_n)$. Cela signifie qu'il ne sera nécessaire de conserver que ce seul «résumé» de l'information originale.
- Les distributions a priori et a posteriori sont de la même forme fonctionnelle. Cette propriété découle directement de l'usage de distributions conjuguées. Elle sera très utile par la suite.

Comme membres notables de la famille exponentielle, on peut retenir les distributions binomiale, multinomiale, gamma, gaussienne, ou de Wishart.

3.2.4 Maximum de vraisemblance, maximum a posteriori

Considérons maintenant l'expression (3.8). Le maximum de vraisemblance θ_{ML} d'un échantillon $\mathbf{X}$ est défini par :

$$\theta_{\mathrm{ML}} = \arg\max_{\theta} p(\mathbf{X}|\theta) = \arg\max_{\theta} \ln p(\mathbf{X}|\theta) \text{ car } \ln(.) \text{ monotone} \tag{3.11}$$

Pour le même échantillon, on définit également le maximum a posteriori θ_{MAP} :

$$\theta_{\mathrm{MAP}} = \arg\max_{\theta} p(\mathbf{X}, \theta) \tag{3.12}$$

Nous voyons immédiatement qu'en utilisant une distribution exponentielle, et en remarquant que $p(\theta|\mathbf{X}) \propto p(\mathbf{X}, \theta)$, ces estimateurs vont être calculés très facilement. En effet, nous disposons alors de l'expression (3.7) pour $p(\theta|\mathbf{X})$, et il suffit d'en extraire le mode. Remarquons qu'un terme a priori a été utilisé pour la définition de (3.7) : on obtient alors θ_{MAP}. Il suffit cependant de remplacer le terme a priori par une constante (i.e. distribution uniforme non-normalisée) pour obtenir θ_{ML}.

3.2.5 Utilisation

Comme précisé dans le paragraphe 3.2.1, l'objectif de l'approche génératrice est de modéliser un échantillon de données. Le modèle estimé peut ensuite être utilisé à diverses fins :

- **Discrimination** : l'échantillon $\mathbf{X}$ sera fourni avec un jeu de labels (ou étiquettes) associé, y définissant ainsi des classes. Chaque classe sera utilisée pour estimer un modèle. Un nouvel individu $\mathbf{x}_{\mathrm{new}}$ sera alors affecté à la classe pour laquelle sa vraisemblance est la plus élevée.
- **Prédiction** : le modèle estimé peut être utilisé pour déterminer :

  $$p(\mathbf{x}_{\mathrm{new}}|\mathbf{X}, \theta) \tag{3.13}$$

  On pourra ainsi évaluer la vraisemblance de $\mathbf{x}_{\mathrm{new}}$ si celui-ci est fourni en entrée, ou échantillonner selon (3.13) pour effectuer une prédiction.
- **Synthèse** : le modèle a posteriori peut être vu comme une représentation synthétique de nos données. Dans un contexte d'apprentissage distribué, par exemple, il suffira donc de ne transmettre que θ au lieu de $\mathbf{X}$. Attention toutefois : en calculant θ_{ML} ou θ_{MAP}, on fait l'hypothèse de données distribuées selon la forme fonctionnelle choisie (i.e. *hypothèse génératrice*). Cette hypothèse peut éventuellement être vérifiée au moyen de tests statistiques classiques. Au cas où elle n'est que faiblement, ou pas vérifiée, la vraisemblance associée sera faible, et résumer notre

échantillon par un maximum de vraisemblance reviendra à perdre une grande partie de l'information.

– **Visualisation** : les *Generative Topographic Maps* sont des mélanges de gaussiennes particuliers destinés à la visualisation d'échantillons avec un support sous-jacent de rang faible [Bishop and Svensen, 1998]. Les mélanges de gaussiennes ont été utilisés en tant qu'enveloppes grossières (*blobs*) pour la caractérisation rapide du type de la scène représentée par une image (e.g. montagne, intérieur, etc) [Schyns and Oliva, 1994, Oliva and Schyns, 1997, Carson et al., 1997, 1999, 2002]. La combinaison d'une hiérarchie de mélanges de gaussiennes [voir Vasconcelos and Lippman, 1998, Garcia and Nielsen, 2010, Garcia et al., 2010], et d'une ACP pour chaque composante, est proposée en tant qu'outil pour la visulisation de données à haute dimensionalité [Wang et al., 2000].

3.3 Mélange de gaussiennes

3.3.1 Motivation

Considérons maintenant K coefficients ω_k tels que $\sum_k \omega_k = 1$, et $0 \leq \omega_k \leq 1 \; \forall k$, ainsi que K distributions f_k, membres de la famille exponentielle, et de forme fonctionnelle identique. On vérifie assez facilement que la fonction suivante est une distribution de probabilité valide :

$$p(\mathbf{x}) = \sum_k \omega_k f_k(\mathbf{x}) \tag{3.14}$$

En tant que membre de la famille exponentielle, la gaussienne multivariée peut être utilisée en lieu et place des f_k dans l'expression (3.14). En général on note $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$ la loi gaussienne de moyenne μ et de matrice de covariance Σ. Relativement au remarques faites au paragraphe 3.2.5, modéliser un échantillon par une seule gaussienne revient à faire une hypothèse assez forte.

Nous écrivons la spécialisation de (3.14), mettant en oeuvre des gaussiennes :

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) \tag{3.15}$$

où ω_k est le poids de la composante k, μ_k est sa moyenne, et Σ_k est sa covariance. Dans ce contexte, on définit les ensembles des poids $\omega = \{\omega_k\}$, des moyennes $\mu = \{\mu_k\}$ et des covariances $\Sigma = \{\Sigma_k\}$. On définit également $\theta_k = \{\omega_k, \mu_k, \Sigma_k\}$. Ainsi $\theta = \{\theta_k\} = \{\omega, \mu, \Sigma\}$.

Avec cette famille de distributions, on ne fait a priori pas d'hypothèse génératrice. En effet, n'importe quel jeu de données peut être modélisé avec une précision arbitraire par un mélange de gaussiennes ; le cas limite où chaque individu d'un échantillon $\mathbf{X}$ est associé avec une gaussienne de moyenne $\mathbf{x}_n$ et de covariance $\Sigma_n \to 0$ le montre bien.

Cependant, s'il démontre le pouvoir modélisant de la famille des mélanges de gaussiennes, ce dernier cas n'a aucun intérêt pratique :

- Aucune synthèse n'est effectuée, dans la mesure où le modèle a alors autant de paramètres que la taille de l'échantillon.
- Si le modèle doit être utilisé à des fins de prédiction, il ne sera capable de générer que l'échantillon utilisé pour l'apprentissage (i.e. il définit la *distribution empirique* de l'échantillon $\mathbf{X}$). Dans le monde réel, un signal ou une information peut être sujette au bruit (erreur de mesure, variabilité naturelle dans une population) qu'on ne saura alors pas du tout capturer.

Pour ces raisons, sur un échantillon de taille N, on cherche en général à estimer un modèle a posteriori à $K << N$ composantes. L'hypothèse génératrice associée (voir 3.2.5) existe alors, tout en restant beaucoup moins forte que pour un modèle à une seule composante.

3.3.2 Variable latente

La somme pondérée de gaussiennes ne fait pas partie de la famille exponentielle. On peut le vérifier en constatant que (3.15) ne vérifie pas la forme générale (3.3). On ne peut donc pas appliquer la formule (3.7) pour obtenir une distribution a posteriori, ou autrement dit il n'existe pas d'estimateur explicite pour un mélange de gaussiennes à K composantes sur un échantillon de taille N.

A chaque individu $\mathbf{x}$, associons une variable latente $\mathbf{z} \in \mathbb{R}^K$ de type «vecteur canonique» (ou *1-parmi-K* dans la littérature [Bishop, 2006, chap. 9]) :

$$\begin{aligned} &\mathbf{z} \in \{\mathbf{e}_1, \ldots, \mathbf{e}_K\}, \text{ avec } \mathbf{e}_k \text{ le } k\text{ème vecteur canonique de } \mathbb{R}^K \\ &\text{et } z_k \text{ la } k\text{ème dimension de } \mathbf{z} \end{aligned} \tag{3.16}$$

On peut interpréter cette variable comme dénotant «l'appartenance» de $\mathbf{x}$ à une composante spécifique. En réalité, un mélange de gaussiennes utilise cette variable de manière implicite (d'où la dénomination *latente*) ; en manipulant (3.15), on obtient en effet :

$$p(\mathbf{x}|\mathbf{z},\theta) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\mu_k,\Sigma_k)^{z_k} \tag{3.17}$$

$$p(\mathbf{z}) = \prod_{k=1}^{K} \omega_k^{z_k} \tag{3.18}$$

Ou encore :

$$p(\mathbf{x}|\mathbf{z}=\mathbf{e}_k,\theta) = p(\mathbf{x}|z_k=1,\theta) = \mathcal{N}(\mathbf{x}|\mu_k,\Sigma_k) \tag{3.19}$$

$$p(\mathbf{z}=\mathbf{e}_k) = p(z_k=1) = \omega_k \tag{3.20}$$

Dans la littérature, cette famille de distributions $p(\mathbf{x},\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ est parfois nommée *famille exponentielle à variables cachées* [Sato, 2001, Smidl and Quinn, 2006]. L'exemple donné ici utilise une variable latente discrète, mais elle pourrait être remplacée par n'importe quelle variable continue, à condition que celle-ci appartienne à la famille exponentielle. Dans la suite de nos développements, nous travaillerons le plus souvent avec des mélanges de gaussiennes. Ce sous-ensemble de la famille exponentielle à variables cachées est en effet le plus souvent utilisé dans la littérature, et dans la prochaine section nous passerons en revue de quelques unes de ses applications.

Par ailleurs, nous avons vu qu'il n'existe pas de solution explicite à l'apprentissage d'un mélange de gaussiennes sur un échantillon : nous aurons donc recours à la recherche d'une solution approchée. Dans la section 3.4, nous présentons l'algorithme EM, la méthode traditionnellement utilisée pour une telle recherche.

3.3.3 Utilisations

Dans ce paragraphe, nous passons en revue quelques unes des utilisations les plus courantes des mélanges de gaussiennes.

Traitement de données multimedia

Les mélanges de gaussiennes peuvent être utilisés pour réaliser la segmentation d'une image selon les niveaux de couleurs [Bishop, 2006, chap. 9]. Considérons maintenant la tâche de chercher des ressemblances parmi une collection d'images. Il existe de nombreuses manières de les caractériser (niveaux de couleur des pixels [Carson et al., 2002, Greenspan et al., 2001], texture, histogrammes de couleurs, descripteurs basés sur l'analyse des gradients d'intensité des pixels (e.g. SIFT [Lowe, 1999]).

Dans [Goldberger et al., 2003], ou encore [Carson et al., 2002], des mélanges de gaussiennes sont utilisés pour modéliser les positions et niveaux de couleurs des pixels d'une image. Cette approche est motivée par le fait qu'il est plus facile de mesurer la similarité entre distributions qu'entre tableaux de pixels, et se sert ainsi des modèles d'images comme base pour un système d'indexation au sein d'une collection d'images.

Les mélanges de gaussiennes sont aussi la méthode standard pour modéliser les locuteurs dans le cadre d'un système de reconnaissance audio [Reynolds et al., 2000].

Fonction de classification

Dans la section **??**, nous avons présenté la tâche de classification non-supervisée ; le mélange de gaussiennes est souvent utilisé à cette fin [Bishop, 2006, chap. 9]. Les variables latentes d'appartenance sont alors interprétées comme une classe (ou label, ou étiquette : la modalité de variable latente $\mathbf{e}_k$ définit l'étiquette de la $k^{\text{ème}}$ classe), et l'apprentissage consistera à inférer les variables latentes et le modèle a posteriori.

Mais il est tout aussi possible :

- d'estimer un modèle de classification supervisée. Cette possibilité est basée sur une formulation probabiliste de la régression linéaire, où la fonction prédictrice de y est posée sous la forme $\mathcal{N}(y|\mathbf{w}^T\mathbf{x}, \epsilon)$. Un mélange de telles distributions est appelé mélange de gaussiennes conditionnel dans la littérature [Bishop, 2006, chap. 14], constituant ainsi une combinaison de fonctions de régression. En transformant la combinaison linéaire de $\mathbf{x}$ avec une fonction sigmoide, on peut en revanche apprendre un modèle de classification [Jordan and Jacobs, 1994].

- de disposer d'une partie des étiquettes réelles ; l'apprentissage d'une partition est alors d'autant plus facile. On cherche alors à inférer les étiquettes manquantes, et le modèle a posteriori adéquat (i.e. approche *cluster-and-label*, voir section 2.4),

- d'effectuer des prédictions sur de nouvelles étiquettes et/ou de nouveaux individus. Le modèle a posteriori est estimé à partir d'un ensemble d'apprentissage $\mathbf{X}$, puis utilisé pour évaluer $p(\mathbf{x}_{\text{new}}|\mathbf{X})$ ou $p(\mathbf{z}_{\text{new}}|\mathbf{x}_{\text{new}}, \mathbf{X})$.

3.4 Algorithme EM

L'algorithme EM est une méthode générale d'inférence de valeurs manquantes et d'estimation de paramètres. Nous allons ici essentiellement rappeler son application à l'estimation d'un mélange de gaussiennes. Mais il en existe diverses variantes et extensions, que nous évoquerons ensuite.

3.4.1 Motivation

Considérons maintenant à nouveau un échantillon $\mathbf{X}$ de taille N, et l'échantillon $\mathbf{Z} \in (\mathbb{R}^K)^N$ associé (supposé inconnu). Posons $\{\mathbf{X}, \mathbf{Z}\}$ comme membre de la famille exponentielle à variables cachées. L'objectif ultime est de trouver le modèle θ_{ML} maximisant la vraisemblance (ou de manière équivalente, la log-vraisemblance) de $\mathbf{X}$, autrement dit :

$$\theta_{\text{ML}} = \arg\max_{\theta} \ln p(\mathbf{X}|\theta) = \arg\max_{\theta} \ln\Big\{\sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta)\Big\} \tag{3.21}$$

Nous commencerons par remarquer que :

- Le terme $\sum_{\mathbf{Z}}$ indique qu'on somme sur les états possibles de $\mathbf{Z}$. Pour une variable continue, il faudrait utiliser une intégration.
- La log-vraisemblance $\mathcal{L}(\mathbf{X}|\theta)$, pour une seule composante gaussienne, est une fonction concave des paramètres. Le maximum de vraisemblance est alors simplement obtenu en extrayant le moment de 1$^{\text{er}}$ ordre de la distribution a posteriori (3.7). En revanche, en tant que log de somme de gaussiennes, (3.21) n'est pas concave en général. Ceci motive la recherche d'une solution approchée.

3.4.2 Principe

Si les *données complètes* $\{\mathbf{X}, \mathbf{Z}\}$ sont issues d'une distribution de la famille exponentielle à variables cachées, alors $\mathcal{L}(\mathbf{X}, \mathbf{Z}|\theta)$ est concave, donc facile à maximiser.

Toutefois, seul $\mathbf{X}$ est connu (ou observé) ; nous ne disposons donc pas d'expression analytique pour le maximum de vraisemblance de $\mathcal{L}(\mathbf{X}, \mathbf{Z}|\theta)$. Un maximum local de cette fonction peut être obtenu grâce à l'algorithme *Expectation-Maximisation* (EM) [Dempster et al., 1977] ; il se trouve que l'estimateur obtenu est également un maximum local de $\mathcal{L}(\mathbf{X}|\theta)$ (preuve dans [Neal and Hinton, 1999], voir paragraphe 3.4.3 pour plus de détails).

EM est un algorithme itératif initialisé par un estimateur $\theta_{\text{old}} = \theta_0$ quelconque, obtenu par exemple de manière aléatoire. Considérons maintenant le cas du mélange de gaussiennes. Connaissant θ_{old}, on peut établir $p(\mathbf{Z}|\mathbf{X}, \theta_{\text{old}})$, qui, en utilisant les expressions (3.17), (3.18) et la règle de Bayes, est donnée par :

$$p(\mathbf{Z}|\mathbf{X}, \theta_{\text{old}}) \propto \prod_{n=1}^{N} \prod_{k=1}^{K} \big[\omega_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)\big]^{z_{nk}} \tag{3.22}$$

C'est *l'étape E* de l'algorithme. Celle-ci consiste à calculer l'espérance des z_{nk} selon (3.22) :

$$\mathbb{E}[z_{nk}] = \frac{\omega_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} \omega_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)} \tag{3.23}$$

En utilisant ces estimateurs, on peut définir $\mathcal{Q}(\theta, \theta_{\text{old}})$, l'espérance de la log-vraisemblance $\mathcal{L}(\mathbf{X}, \mathbf{Z}|\theta)$:

$$\mathcal{Q}(\theta, \theta_{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta_{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\theta) \tag{3.24}$$

Comme nous l'avons écrit en début de paragraphe, la maximisation de cette quantité selon θ est aisée : on a alors une fonction concave dont il suffit de calculer la différentielle. Un estimateur du modèle mis à jour est ainsi obtenu selon :

$$\theta_{\text{new}} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta_{\text{old}}) \tag{3.25}$$

C'est *l'étape M* de l'algorithme. θ_{new} remplace alors θ_{old}, et les étapes E et M sont ainsi itérées jusqu'à convergence (voir paragraphe 3.4.3 pour des détails à ce sujet). Dans le cas du mélange de gaussiennes, l'application de la formule (3.25) permet de trouver :

$$\omega_k^{\text{new}} = \frac{N_k}{N} \tag{3.26}$$

$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_n \mathbb{E}[z_{nk}] \mathbf{x}_n \tag{3.27}$$

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_n \mathbb{E}[z_{nk}] (\mathbf{x}_n - \mu_k^{\text{new}})(\mathbf{x}_n - \mu_k^{\text{new}})^T \tag{3.28}$$

$$\text{avec } N_k = \sum_n \mathbb{E}[z_{nk}] \tag{3.29}$$

Les démonstrations détaillées des formules présentées dans ce paragraphe peuvent être trouvées dans [Bishop, 2006, chap. 9].

Il est possible de trouver une équivalence entre cet algorithme et une descente de gradients [Titterington, 1984]. Par ailleurs, il a été montré qu'en imposant des matrices de covariance isotropiques (i.e. $c\mathbf{I}$) et identiques pour toutes les composantes, l'algorithme EM pour le mélange de gaussiennes était équivalent à l'algorithme K-means [McQueen, 1967].

3.4.3 Propriétés

Nous montrons maintenant que chaque itération de l'algorithme EM augmente la fonction de vraisemblance mentionnée dans l'équation (3.21). Sous condition d'existence d'une borne supérieure pour cette fonction, cette propriété sera alors suffisante pour dire que si l'algorithme converge, alors on obtient un maximum de vraisemblance local pour θ.

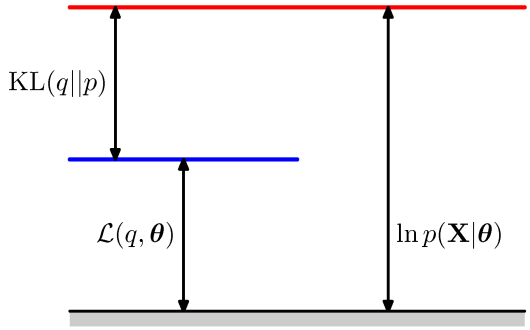


FIG. 3.1 – Décomposition de la log-vraisemblance en borne inférieure et divergence KL. (Extrait de [Bishop, 2006])

Soit une distribution quelconque $q(\mathbf{Z})$. On l'utilise pour décomposer la fonction de vraisemblance à optimiser :

$$\mathcal{L}(\mathbf{X}|\theta) = \mathcal{L}(q, \theta) + \mathrm{KL}(q||p) \tag{3.30}$$

$$\text{avec } \mathcal{L}(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z}|\theta)}{q(\mathbf{Z})} \tag{3.31}$$

$$\text{et } \mathrm{KL}(q||p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z}|\mathbf{X}, \theta)}{q(\mathbf{Z})} \tag{3.32}$$

$\mathrm{KL}(q||p)$ est la divergence de Kullback-Leibler de $p(\mathbf{Z}|\mathbf{X}, \theta)$ par rapport à $q(\mathbf{Z})$. La divergence KL est positive ou nulle, non-bornée, croissant selon la différence entre deux distributions de probabilité. Ce n'est cependant ni une métrique, ni une distance.

Nous avons utilisé la notation habituellement dévolue à la log-vraisemblance pour l'expression (3.31). Toutefois, comme $KL(.||.) \geq 0$, nous désignerons le plus souvent cette fonction particulière comme la *borne inférieure* à la log-vraisemblance. Cette décomposition est illustrée sur la figure 3.1.

Etape E

Considérons maintenant θ fixé à θ_{old}. Nous savons que :

- $\mathcal{L}(\mathbf{X}|\theta_{\text{old}})$ est constante à cette étape, car elle ne dépend pas de q,
- $\mathrm{KL}(q||p) = 0$ ssi $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta_{\text{old}})$,

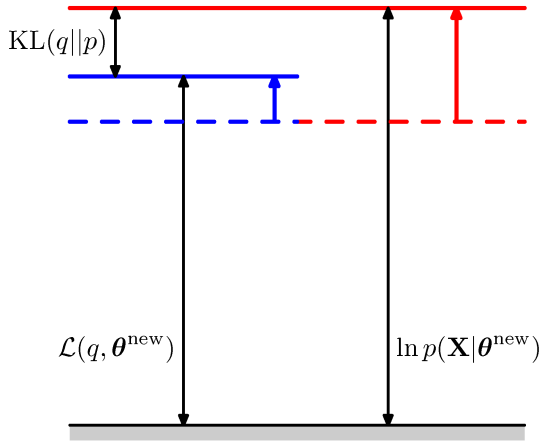


FIG. 3.2 – Conséquences de l'augmentation de la borne inférieure $\mathcal{L}(q, \theta)$ à l'étape M. (Extrait de [Bishop, 2006])

donc $\mathcal{L}(q, \theta_{\text{old}})$ sera maximisée pour $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta_{\text{old}})$. On retrouve bien l'étape E évoquée dans le paragraphe 3.4.2.

Etape M

Cette fois ci, considérons q fixée. De l'étape E précédente, nous savons que $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta_{\text{old}})$. On peut donc substituer dans (3.31), ce qui donne :

$$\mathcal{L}(q, \theta) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta_{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\theta) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta_{\text{old}}) \ln p(\mathbf{Z}|\mathbf{X}, \theta_{\text{old}}) \tag{3.33}$$

$$\mathcal{L}(q, \theta) = \mathcal{Q}(\theta, \theta_{\text{old}}) + \text{ const} \tag{3.34}$$

Synthèse

La maximisation de $\mathcal{Q}$ selon θ entraine donc nécessairement une augmentation de $\mathcal{L}(q, \theta)$. En général, $\theta_{\text{new}} \neq \theta_{\text{old}}$, ce qui cause également une augmentation du terme de divergence KL dans l'équation (3.30).

Ces augmentations cumulées conduisent nécessairement à une augmentation de $\mathcal{L}(\mathbf{X}|\theta)$ (voir figure 3.2). Nous avions précédemment précisé que l'étape E laissait cette valeur inchangée ; une succession d'étapes E et M augmente donc monotoniquement la log-vraisemblance du modèle estimé. Ce lien entre l'algorithme EM et l'augmentation monotone d'une borne inférieure à la log-vraisemblance a été mis en lumière dans [Neal and Hinton, 1999]. En cas de convergence, on a alors bien un maximum local.

Sur la figure 3.3, on voit bien que l'optimisation directe de $\mathcal{L}(\mathbf{X}|\theta)$ serait difficile. La fonction concave $\mathcal{L}(q, \theta)$ est donc utilisée de la manière suivante :

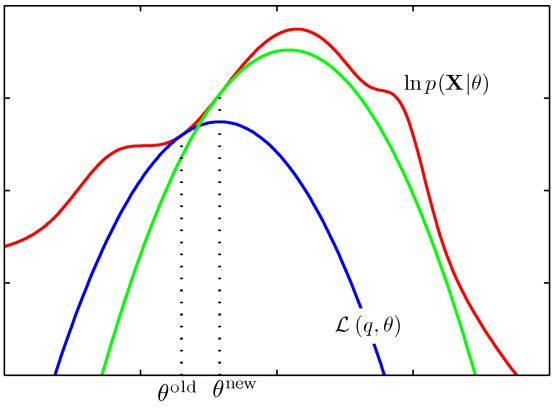


FIG. 3.3 – Exemple de mise en oeuvre d'EM, avec θ en abscisse. (Extrait de [Bishop, 2006])

- A l'étape E, θ_{old} est fixé, et $\mathcal{L}(q, \theta_{\text{old}})$ est positionnée de manière tangente à $\mathcal{L}(\mathbf{X}|\theta_{\text{old}})$. En effet, $KL(q||p)$ est nulle en ce point, et c'est alors son minimum ; son gradient selon θ y est donc nul, ce qui implique que les gradients de la log-vraisemblance de $\mathbf{X}$ et de la borne inférieure sont égaux.
- A l'étape M, on trouve assez simplement arg max d'une fonction concave. Le paramètre θ_{new} résultant est alors utilisé pour l'étape E suivante.

Dans cette section, l'algorithme EM a été essentiellement vu comme une méthode pour maximiser une borne inférieure à une fonction de vraisemblance. Cette formulation nous sera utile pour notre introduction de l'apprentissage variationnel bayésien.

Rappelons que les relations établies de cet algorithme avec K-means, et ainsi de manière plus générale avec les problèmes de quantification vectorielle (e.g. [Martinetz et al., 1993]), peuvent aussi permettre de voir l'objectif d'EM comme la minimisation d'une fonction de distorsion.

3.4.4 Limitations

- **Permutations** : L'algorithme EM permet d'extraire un modèle maximisant localement la fonction de vraisemblance. Celle-ci, contrairement au cas simple présenté sur la figure 3.3, est le plus souvent multimodale ; en effet, un mélange à K composantes peut-être permuté de $K!$ manières différentes. Il y aura donc autant de modes que de permutations possibles. Si on n'attribue pas de sémantique a priori à nos composantes, cela ne pose pas de problème ; l'algorithme nous fournira une solution parmi $K!$ également valides.

- **Sensibilité à l'initialisation** : L'initialisation décide d'une «région» de la fonction de vraisemblance dont l'algorithme EM fait une exploration locale. Parmi les

maxima associés à différentes régions, certains sont potentiellement meilleurs que d'autres. On doit donc choisir une initialisation conduisant le plus sûrement possible vers les meilleures solutions. Il n'existe pas de choix systématique, aussi des heuristiques ont été proposées. Par exemple, Biernacki et al. [Biernacki et al., 2003] proposent d'utiliser un «pool» d'initialisations aléatoires, sur lesquelles seules quelques étapes d'EM sont réalisées. Les estimations partielles associées aux meilleures valeurs de vraisemblance sont alors conservées. De manière générale, selon l'initialisation on peut s'orienter vers de «mauvais» maxima locaux [McLachlan and D., 2000, chap. 3.10]. Nous en donnerons un exemple au travers des expériences faites section 3.5.3, et nous montrerons alors une approche permettant de les éviter.

– **Fonction de vraisemblance non-bornée** : dans la section 3.4.3, nous avions précisé que l'algorithme EM converge ssi la fonction de vraisemblance est bornée. Dans le cas du mélange de gaussiennes, les itérations peuvent nous conduire à ajuster une composante sur un seul point. Dans ce cas, en supposant l'utilisation de gaussiennes isotropiques (le raisonnement reste valable dans le cas général), la contribution à la vraisemblance associée sera alors $\mathcal{N}(\mathbf{x}_n|\mathbf{x}_n, \sigma\mathbf{I}) \propto \frac{1}{\sigma}$. La fonction n'est donc pas bornée. Toutefois ce problème est bien caractérisé, i.e. il n'arrive que quand une composante «s'écroule» sur un point. Il est donc assez facile à détecter. Celeux propose notamment de contraindre les matrices de covariance du modèle à un déterminant constant [Biernacki et al., 2003].

– **Complexité du modèle** : dans cette section, nous avons supposé la complexité du modèle (i.e. le nombre K de composantes) connue a priori. Cependant, la vraisemblance est une fonction croissante de cette complexité (avec le cas extrême où chaque point est associé à une composante de vraisemblance tendant vers l'infini. Ce choix n'est donc pas trivial ni systématique ; il traduit plutôt une notion de compromis (parfois connue sous le nom de principe du *rasoir d'Occam*). En résumé, une bonne solution se doit d'être la plus simple possible tout en restant acceptable. En d'autres termes, la complexité doit être pénalisée, afin d'orienter l'algorithme vers les meilleurs compromis. Ce principe se rapproche de la régularisation effectuée dans le cadre des algorithmes de régression ; afin d'éviter le sur-apprentissage, on cherche à limiter la magnitude des coefficients de régression. Dans le paragraphe 3.4.5, nous verrons quelques extensions classiques permettant de traiter ce problème.

3.4.5 Variantes et extensions

Régularisation des solutions

En régularisant les estimateurs de notre modèle, on entend résoudre le problème lié à notre fonction de vraisemblance non-bornée : si on ne borne pas celle-ci explicitement, tout au moins peut-on s'assurer qu'aucune de nos composantes s'écroule, ce qui revient à détecter si l'estimateur de covariance actuel est numériquement singulier ou non.

Un moyen simple pour réaliser cette détection, souvent utilisé dans les logiciels statistiques, est proposé lignes 12 et 13 de l'algorithme 1. La composante associée est alors retirée du modèle.

Toutefois, cette composante modélisait un nombre non-nul d'éléments, avec une précision élevée : cela équivaut à une valeur de vraisemblance conséquente. Or la fin de l'algorithme est décidée par la convergence de la vraisemblance du modèle, une fonction en principe monotone croissante. La suppression d'une composante singulière cause en général une chute de la vraisemblance, donc la fin précipitée de l'algorithme. Pour éviter cela, il faut alors réinitialiser la valeur de référence servant à décider de la convergence.

D'un point de vue algorithmique, à l'instar de l'algorithme K-means, il est aussi possible, avec EM, qu'après un certain nombre d'itérations une composante n'ait plus aucun élément qui lui soit affecté : il faut alors ne plus la prendre en compte. Cette opération n'a en revanche aucune conséquence sur le critère de convergence.

A la lumière de ces remarques, nous proposons l'algorithme 1 en tant qu'implémentation approchée mais stable d'EM. Cette implémentation est utilisée dans nos expériences.

```
   Entrées : Echantillon X
   Sorties : Modèle a posteriori θ
 1 Initialiser θ de manière aléatoire ;
 2 likelihood ← −∞;
 3 Δlikelihood ← ∞ ;
 4 tant que Δlikelihood > unSeuil faire
 5 |   Etape E ;
 6 |   Etape M ;
 7 |   pour tous les θ_k ∈ θ faire
 8 |   |   si N_k = 0 alors
 9 |   |   |   Supprimer θ_k ;
10 |   |   fin
11 |   |   sinon
12 |   |   |   {λ_i} ← valeurs propres ordonnées de Σ_k ;
13 |   |   |   si λ_d/λ_1 < unAutreSeuil alors
14 |   |   |   |   Supprimer θ_k ;
15 |   |   |   |   likelihood ← −∞;
16 |   |   |   |   Δlikelihood ← ∞ ;
17 |   |   |   fin
18 |   |   fin
19 |   fin
20 |   newLikelihood ← calcul vraisemblance de θ ;
21 |   Δlikelihood ← newLikelihood − likelihood ;
22 |   likelihood ← newLikelihood ;
23 fin
```

Algorithme 1 : Algorithme EM stable

Choix de la complexité

Caractérisé dans le paragraphe précédent, ce problème est classiquement résolu en utilisant des critères de pénalité (e.g. AIC [Akaike, 1974], BIC [Schwarz, 1978]). En résumé, ceux-ci attribuent une «note» à chaque modèle. Il faut alors entrainer plusieurs modèles, de complexités variées (i.e. pour le mélange de gaussiennes, différentes valeurs de K), et le critère sert à choisir le meilleur parmi une population de candidats.

Nous avons ici une incarnation de l'apprentissage bayésien : l'apprentissage doit limiter au maximum la taille du modèle appris. Dans la prochaine section, nous verrons des approches plus sophistiquées pour implémenter ce principe. Notons que de nombreuses variantes des critères AIC et BIC ont été étudiées dans la littérature ; ainsi, Xiang et Gong ont présenté quelques variantes spécialement adaptées à l'apprentissage de modèles de mélanges [Xiang and Gong, 2006]. Dans le contexte d'une classification réalisée par des cartes de Kohonen [Kohonen, 1990], une technique de regroupement de

centroïdes a été proposée [Cabanes and Bennani, 2008]. En particulier, les centroides sont regroupés s'ils sont séparés par des zones de forte densité dans l'échantillon. Cette information de densité est extraite grâce à la structure topologique des cartes de Kohonen.

Il est aussi possible d'adjoindre des étapes d'éclatement (*split*) et de fusion (*merge*) au étapes habituelles E et M [Ueda et al., 2000]. L'éclatement survient quand une composante modélise trop faiblement la distribution empirique de ses données locales, et la fusion quand deux composantes ont un recouvrement trop important. Cette approche ne permet pas à proprement parler de réguler la complexité, mais permet d'éviter certains des maxima locaux évoqués dans [McLachlan and D., 2000, chap. 3.10].

Passage à l'échelle

Celeux et Govaert ont montré que, sous certaines hypothèses, l'algorithme EM a une vitesse de convergence linéaire [Celeux and Govaert, 1992], notamment justifiée par l'équivalence d'EM à un algorithme de descente de gradients. Ils précisent tout de même que la vitesse observée est susceptible d'être assez fortement dépendante à l'initialisation de l'algorithme, ainsi qu'aux données elles-mêmes.

La plupart des variantes d'EM permettant d'accélérer la convergence se basent sur le principe d'*EM généralisé* [Bishop, 2006, chap. 9] : la convergence reste assurée si chaque succession d'étapes E et M augmente la vraisemblance sans forcément la maximiser.

Ainsi, Neal et Hinton ont proposé des variantes incrémentales significativement plus rapides [Neal and Hinton, 1999]. Celeux et al. ont étudié la convergence d'un algorithme mettant à jour chaque composante séparément, tout en s'assurant que la fonction optimisée est équivalente [Celeux et al., 2001]. Jordan et Jacobs ont proposé un algorithme de classification reposant sur une hiérarchie de modèles linéaires généralisés apprise via une variante d'algorithme EM. Ils ont alors montré que leur approche était ainsi nettement plus rapide que les méthodes de l'état de l'art de ce domaine [Jordan and Jacobs, 1994].

Représentation graphique du mélange de gaussiennes

Dans la section précédente, nous avons présenté le mélange de gaussiennes en tant qu'instance de la famille exponentielle à variables cachées. Nous proposons ici un autre formalisme, inspiré des réseaux bayésiens : le DAG (*Directed Acyclic Graph*), ou graphe orienté acyclique. Par exemple, le mélange de gaussiennes pourrait être résumé de manière graphique par la figure 3.4.

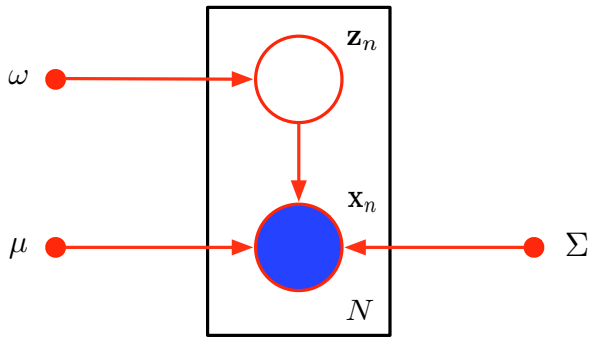


FIG. 3.4 – Représentation du mélange de gaussiennes par un DAG

Le formalisme associé est le suivant :

- toute variable issue d'une distribution de probabilité est représentée par un grand cercle,
- tout paramètre ajustable est représenté par un petit cercle,
- une flèche orientée indique une dépendance conditionnelle,
- l'intérieur d'un rectangle dénote une ou plusieurs variables i.i.d, avec la taille de la population (N sur la figure 3.4),
- un grand cercle vide indique une distribution inconnue (e.g. distribution sur un paramètre à estimer, variable latente). Un grand cercle plein indique que cette variable est observée,
- par analogie avec la terminologie des graphes, les grands cercles (vides ou pleins) sont parfois appelés *noeuds*.

En réalité, tout DAG dont chaque variable aléatoire (i.e. grand cercle plein ou vide) est de la famille exponentielle représente une distribution jointe de la famille exponentielle à variables cachées. Ainsi, pour tout modèle respectant ce formalisme, il sera possible de décliner un algorithme EM adapté. Notons bien cette existence s'accompagne des mêmes limitations.

Apprentissage de chaines de Markov cachées (HMM)

Les HMM peuvent être représentés par un DAG (voir figure 3.5). Dans ce modèle, les variables observées ne sont pas i.i.d ; on parle de données séquentielles, et un indice indique l'ordre d'arrivée. L'hypothèse de Markov consiste à réduire la dépendance entre données observées à celle les précédant immédiatement dans la séquence.

Le modèle de la chaine de Markov cachée consiste à considérer que les données observées $\mathbf{x}_n$ sont *émises* par un *état* inconnu $\mathbf{z}_n$; l'état $\mathbf{z}_n$ est produit par *transition*

depuis l'état $\mathbf{z}_{n-1}$.

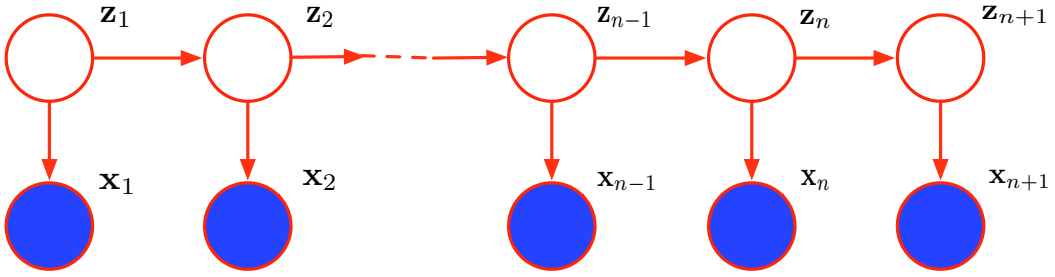


FIG. 3.5 – Représentation d'un chaine de Markov cachée par un DAG

En général, des distributions de la famille exponentielle sont utilisées pour modéliser les transitions et les émissions ; dans ce cas, une variante d'EM, l'algorithme de Baum-Welch [Welch, 2003], peut être utilisé afin d'apprendre les paramètres de ces distributions.

3.5 Apprentissage variationnel

Dans cette section, nous présentons l'apprentissage variationnel de modèles de mélange. En effet, cette technique a servi de fondement pour certaines de nos contributions, et l'exposition de ses mécanismes en aidera la compréhension.

Nous commençons par exposer les principes de l'apprentissage variationnel en tant qu'implémentation du principe bayésien, et nous donnerons un algorithme générique. Cette approche, déclinée parfois sous des vocables distincts, mais complémentaires, peut être reliée à l'algorithme EM présenté dans la section précédente.

Nous motivons l'utilisation de cette approche par le fait qu'elle permet de gérer de manière élégante la plupart des problèmes évoqués au paragraphe 3.4.4. L'estimation de mélanges de gaussiennes et d'ACP probabilistes au moyen de cette méthode servent à illustrer les propos. Nous présentons enfin quelques variantes et alternatives à cette technique d'apprentissage dans notre contexte.

3.5.1 Apprentissage bayésien et principe variationnel

Dans la section précédente, nous avons vu que la maximisation d'une fonction de vraisemblance amenait différents problèmes, dont une impossibilité de décider de la complexité du modèle à estimer. L'apprentissage bayésien, que nous avons déjà brièvement

évoqué aux paragraphes 3.2.3 et 3.4.5, est une manière rigoureuse de pallier ce problème.

Formellement, reprenant au passage la terminologie de l'équation (3.9), au lieu de maximiser directement $p(\mathbf{X}|\theta)$, nous chercherons à maximiser la vraisemblance complète :

$$p(\mathbf{X}, \theta) = p(\theta)p(\mathbf{X}|\theta) \tag{3.35}$$

pour un échantillon $\mathbf{X}$ et un modèle θ quelconques. Nous voyons que cette expression générale incorpore la notion de «compromis» (ou rasoir d'Occam) ; le paramètre θ doit être vraisemblable selon les données, mais aussi selon une information a priori matérialisée par $p(\theta)$. Celle-ci peut être :

- une distribution n'indiquant pas de préférence particulière, ou non-informative, agissant alors uniquement à des fins de régularisation. La distribution impropre de Jeffreys [Jeffreys, 1961] en est un exemple.
- une distribution indiquant des propriétés recherchées pour le modèle cible. Par exemple, dans le paragraphe 3.5.2, nous utiliserons une distribution de Dirichlet afin de restreindre la complexité du modèle recherché.

Nous avons déjà vu que l'apprentissage du modèle a posteriori $p(\theta|\mathbf{X})$ n'est pas trivial pour des familles de distributions compliquées (e.g. mélanges de gaussiennes). Dans le cas bayésien, la solution la plus immédiate est la recherche d'un estimateur θ_{MAP}, rapidement présenté paragraphe 3.2.4, par une variante de l'algorithme EM.

Toutefois, si cette approche prend bien en compte une information a priori, elle n'utilise que le mode d'une approximation à la distribution a posteriori sur les paramètres, et, dans des cas extrêmes, peut conduire à des dégénérescences algorithmiques au même titre que l'algorithme EM classique [Beal, 2003]. L'approche variationnelle incorpore également une distribution a priori, mais réalise plutôt une optimisation dans l'espace des fonctions.

Plus précisément, on va considérer le vrai modèle a posteriori (i.e. qui serait obtenu par intégration directe sur la distribution jointe des données et des paramètres) comme inconnu et impossible à trouver directement ou via une méthode itérative simple, et en rechercher une approximation parmi une famille de fonctions plus simples. Pour cela, supposons que le modèle θ peut être partitionné en q éléments :

$$\theta = [\theta_1^T, \theta_2^T, \ldots, \theta_q^T]^T \tag{3.36}$$

L'a priori $p(\theta)$ étant alors partitionné de même. On utilise cette caractéristique pour définir une famille de distributions approchées $\breve{p}(\theta|\mathbf{X})$ comme suit :

$$\breve{p}(\theta|\mathbf{X}) = \prod_{i=1}^{q} \breve{p}(\theta_i|\mathbf{X}) \tag{3.37}$$

Alors le théorème suivant est valable [Smidl and Quinn, 2006] :

Théorème 3.5.1 *La distribution approchée vérifiant :*

$$\tilde{p}(\theta|\mathbf{X}) = \arg\min_{\breve{p}(\theta|\mathbf{X})} KL\Big(\breve{p}(\theta|\mathbf{X})||p(\theta|\mathbf{X})\Big) \tag{3.38}$$

est obtenue pour :

$$\tilde{p}(\theta_i|\mathbf{X}) \propto \exp\Big(\mathbb{E}_{\tilde{p}(\theta_{/i}|\mathbf{X})}[\mathcal{L}(\mathbf{X},\theta)]\Big),\ i = 1,\ldots,q \tag{3.39}$$

ou, de manière équivalente, (3.40)

$$\ln \tilde{p}(\theta_i|\mathbf{X}) = \mathbb{E}_{\tilde{p}(\theta_{/i}|\mathbf{X})}[\mathcal{L}(\mathbf{X},\theta)] + const,\ i = 1,\ldots,q \tag{3.41}$$

avec $\theta_{/i}$ le complémentaire de θ_i dans θ, et $\tilde{p}(\theta_{/i}|\mathbf{X}) = \prod_{j=1,j\neq i}^{q} \tilde{p}(\theta_j|\mathbf{X})$

Ce théorème nous fournit une méthode systématique pour l'obtention de la meilleure distribution approchée au modèle a posteriori exact mais inconnu. En effet, en initialisant (par exemple aléatoirement) $q-1$ composantes de θ, et en itérant la mise à jour de chaque estimateur selon (3.41) jusqu'à convergence, on obtient l'approximation la plus proche du modèle a posteriori au sens de la divergence KL.

La convergence de cette classe d'algorithmes a été prouvée par Sato [Sato, 2001]. On remarquera toutefois qu'il n'existe pas de garantie d'unicité ou d'existence d'un minimiseur exact parmi l'espace des distributions approchées.

Cet algorithme est basé sur la minimisation d'une divergence KL ; nous avons vu dans le paragraphe 3.4.3 que cela caractérise l'algorithme EM. Certains auteurs ont utilisé ce formalisme pour aboutir au théorème 3.5.1 [Jordan et al., 1999, Attias, 2000, Beal, 2003], [Bishop, 2006, chap. 10], présentant alors les algorithmes en découlant comme *pseudo-EM* ou *EM variationnels*. Nous utiliserons souvent cette terminologie par la suite.

Dans la même littérature, la famille $\breve{p}(\theta|\mathbf{X})$ est nommée *distribution variationnelle*, dans la mesure où celle-ci est une fonctionnelle amenée à varier au cours de l'exécution de l'algorithme. De manière homogène à la discussion du paragraphe 3.4.3, elle est souvent notée $q(\theta)$, le conditionnement par les données étant posé de manière implicite. Les

facteurs $\breve{p}(\theta_i|\mathbf{X})$ sont alors appelés *termes variationnels*, et notés $q(\theta_i)$. Les distributions optimales $\tilde{p}(.|\mathbf{X})$, obtenues in fine par l'exécution de l'algorithme, sont notées $q^*(.)$.

Notons que notre algorithme est de la même nature qu'EM, ce qui signifie que le maximum obtenu est local. Toutefois, comme ce maximum est déterminé par une optimisation dans un espace de fonctions au lieu d'un espace de valeurs, cela permet de garantir une certaine régularité à nos solutions. Celles-ci sont trouvées parmi un continuum de fonctions respectant au mieux les propriétés recherchées (décrites notamment dans [Xiang and Gong, 2006]).

Nous illustrons cette nuance avec le cas du mélange de gaussiennes dans le paragraphe suivant. Nous montrons notamment, résultats expérimentaux à l'appui, que l'approche variationnelle permet de traiter de manière rigoureuse le problème de surapprentissage inhérent au mélange de gaussiennes.

Les approches variationnelles dérivées d'EM [Jordan et al., 1999, Attias, 2000, Beal, 2003], [Bishop, 2006, chap. 10] présentent un point de vue intéressant sur le plan algorithmique ; en effet elles démontrent qu'en appliquant la formule (3.41), on maximise une borne inférieure à la log-vraisemblance marginale (constante, voir paragraphe 3.2.3) $\mathcal{L}(\mathbf{X}) = \int \mathcal{L}(\mathbf{X}, \theta) d\theta$. La convergence est donc désormais garantie en toute circonstance. Cette borne inférieure $\mathcal{L}(q)$ est donnée par :

$$\mathcal{L}(q) = \int q(\theta) \ln\left\{ \frac{p(\mathbf{X}, \theta)}{q(\theta)} \right\} \tag{3.42}$$

Elle est parfois nommée *énergie libre* [Sato, 2001], et notée $\mathcal{F}$, par opposition à $\mathcal{L}$, la vraisemblance marginale [Beal, 2003]. Nous remarquerons que la formule (3.42) rappelle fortement la formule (3.31), présentée au cours de notre discussion sur l'interprétation d'EM. En réalité, (3.42) est une généralisation de (3.31), où toute variable non-observée (latente ou paramètre) est intégrée dans le terme général θ.

Cette valeur peut-être utilisée pour contrôler la convergence de l'algorithme, même si, comme nous le verrons section 3.5.2, d'autres moyens peuvent être efficacement employés.

3.5.2 Exemple du mélange de gaussiennes

Dans le paragraphe précédent nous avons présenté les principes et propriétés de l'algorithme EM variationnel dans le cas général. Nous en présentons maintenant l'implémentation appliquée au mélange de gaussiennes.

Nous verrons notamment que l'utilisation de ce membre de la famille exponentielle à variables cachées rend l'implémentation de l'algorithme EM variationnel particulièrement aisée. Rappelons que désormais, comme précisé à la fin du paragraphe précédent, les variables latentes $\mathbf{Z}$ sont intégrées dans θ. Ce paragraphe est essentiellement inspiré de Bishop [Bishop, 2006, chap. 10].

Modèle de vraisemblance

Nous avons vu au paragraphe 3.5.1 qu'au préalable de l'application de l'algorithme EM variationnel, il est nécessaire de définir une fonction de vraisemblance jointe (parfois appelée modèle d'observation [Smidl and Quinn, 2006]) pour un mélange de gaussiennes à K composantes.

On commence par rappeler la vraisemblance donnée par les formules (3.17) et (3.18), mises à l'échelle d'un échantillon de taille N :

$$p(\mathbf{X}|\mathbf{Z}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)^{z_{nk}} \tag{3.43}$$

$$p(\mathbf{Z}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\omega_k^{z_{nk}} \tag{3.44}$$

Dans le paragraphe 3.5.1, nous avons vu que toute variable latente ou paramètre était traitée indifféremment comme une variable aléatoire. De même nous avons évoqué une vraisemblance a priori pour les paramètres du modèle. Les distributions (3.43) et (3.44) faisant partie de la famille exponentielle, le choix naturel pour l'a priori est l'utilisation des distributions conjuguées associées, dont l'existence a été prouvée au paragraphe 3.2.3.

La variable latente $\mathbf{Z}$ est un produit de multinomiales i.i.d conditionnées par les poids ω_k ; l'a priori sur les poids est donc une distribution de Dirichlet, la distribution conjuguée de la multinomiale :

$$p(\omega) = \mathrm{Dir}(\omega|\alpha_0) = \frac{\Gamma(\sum_k \alpha_{0k})}{\prod_{k=1}^{K}\Gamma(\alpha_{0k})}\prod_{k=1}^{K}\omega_k^{\alpha_{0k}-1} \tag{3.45}$$

Nous pouvons constater que la distribution a priori sur les paramètres ω_k est elle-même conditionnée par des paramètres $\alpha_0 = \{\alpha_{0k}\}$. On parle alors d'*hyper-paramètres*. La choix de α_0 décidera notamment des propriétés (ou encore de la régularité) que doit respecter notre solution.

$\mathbf{X}$ conditionnée par $\mathbf{Z}$ est distribuée selon des gaussiennes de moyennes μ_k et de matrices de covariance Σ_k. De manière équivalente, et dans un souci de simplification des développements ultérieurs, nous utiliserons des matrices de précision (i.e. covariance inverse) Λ en lieu et place des matrices de covariance Σ. La distribution conjuguée jointe pour les moyennes et précisions est alors :

$$p(\mu, \Lambda) = p(\mu|\Lambda)p(\Lambda) = \prod_k \mathcal{N}\big(\mu_k|\mathbf{m}_{0k}, (\beta_{0k}\Lambda_k)^{-1}\big)\mathcal{W}\big(\Lambda_k|\mathbf{W}_{0k}, \nu_{0k}\big) \tag{3.46}$$

où $\mathcal{W}(.|.,.)$ est la distribution de Wishart. Notons que nous avons défini un nouvel ensemble d'hyper-paramètres, $\{\mathbf{m}_{0k}, \mathbf{W}_{0k}, \beta_{0k}, \nu_{0k}\}$. Aucune hypothèse particulière n'a faite dans la définition de nos distributions a priori ; en effet, dans les expressions (3.43) et (3.44), ω et le couple (μ, Λ) ne sont pas inter-dépendants.

Nous obtenons ainsi le modèle de log-vraisemblance complète, préalable à l'application de l'algorithme variationnel :

$$\begin{aligned} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \omega, \mu, \Lambda) &= \mathcal{L}(\theta) + \mathcal{L}(\mathbf{Z}|\theta) + \mathcal{L}(\mathbf{X}|\mathbf{Z}, \theta) && (3.47) \\ &= \ln p(\omega) + \ln p(\mu, \Lambda) + \ln p(\mathbf{Z}) + \ln p(\mathbf{X}|\mathbf{Z}) && (3.48) \end{aligned}$$

Ce modèle est illustré par un DAG sur la figure 3.6.

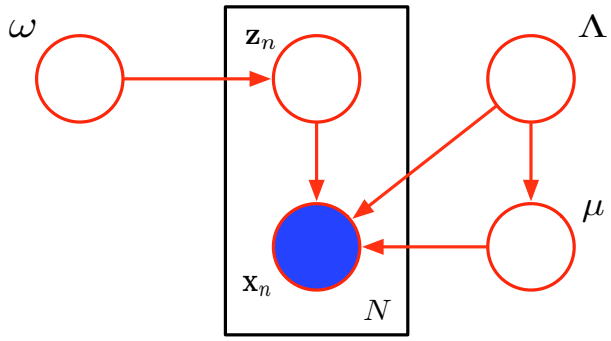


FIG. 3.6 – Représentation du mélange de gaussiennes complet par un DAG

Paramétrage

Dans l'expression (3.48), le «compromis» bayésien est illustré ; en particulier, le choix des étiquettes a posteriori $\mathbf{Z}$ doit, à la fois, aboutir à un modèle vraisemblable

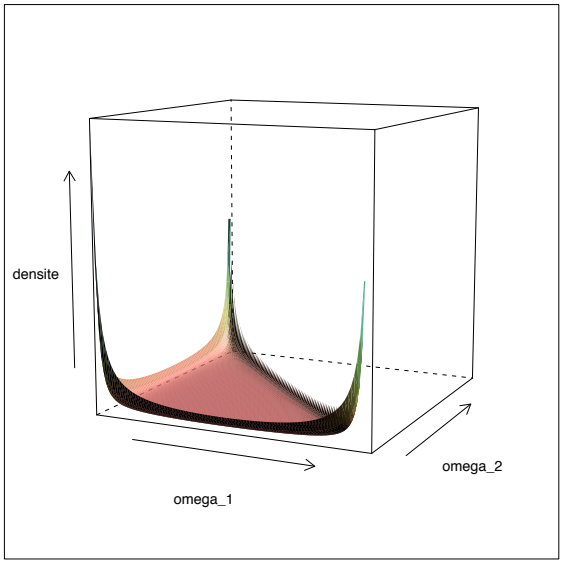


FIG. 3.7 – densité de probabilité de la distribution de Dirichlet pour $k = 3$, et $\alpha_{0k} = 0.1$. Les valeurs ω_k sont comprises dans le simplexe $\omega_k \in [0, 1]\ \forall k$, $\sum_k \omega_k = 1$.

(terme $\ln p(\mathbf{Z})$), et rester proche de l'a priori sur les poids (terme $\ln p(\omega)$). Le choix de la complexité pour le modèle a posteriori (ou autrement dit le choix du nombre de composantes K) peut donc être intégré en choisissant un paramètre α_0 encourageant le moins de composantes «actives» possible.

Bishop préconise l'utilisation de $\alpha_{0k} = 10^{-3}\ \forall k$. Cette valeur peut être justifiée en constatant que pour $\alpha_{0k} < 1$ les modes d'une distribution de Dirichlet $p(\omega|\alpha_0)$ sont localisés aux ω annulant au moins un des ω_k (voir figure 3.7). Ainsi, avec cet a priori, notre algorithme variationnel cherchera à annuler le maximum de coefficients parmi un ensemble de K coefficients ω.

Une fois l'estimation terminée, seules $K' < K$ composantes du mélange de gaussiennes seront donc encore actives ; il suffit alors d'éliminer les composantes ne jouant aucun rôle du modèle a posteriori. Conséquemment, l'algorithme doit être initialisé avec un K arbitrairement grand, dont on doit être sûr qu'il sera supérieur au nombre de composantes idéal, finalement retenu.

En revanche, on n'a pas de connaissance a priori sur la localisation ou la forme des différentes composantes gaussiennes ; on peut donc répartir aléatoirement les moyennes a priori $\mathbf{m}_{0k}$ dans le domaine des données. Les précisions sont choisies diagonales, et à l'échelle du domaine des données. Les hyper-paramètres β servent à fins de normalisation, on peut donc arbitrairement choisir $\beta_{0k} = 1,\ \forall k$. Devant refléter un échantillon

de taille faible, les degrés de liberté a priori sont fixés à leur plus petite valeur possible, $\nu_{0k} = d + 1$.

Nous définissons maintenant la distribution variationnelle à optimiser. On notera qu'à la différence de l'algorithme EM décliné dans la section 3.4, les variables latentes, étant inconnues, sont traitées au même niveau que les paramètres du modèle à estimer (voir fin de paragraphe 3.5.1). Par définition, la distribution variationnelle suit la même factorisation que l'a priori et la vraisemblance (voir paragraphe 3.5.1). Nous avons donc :

$$q(\theta) = q(\omega).\prod_{k=1}^{K}\left(q(\mu_k, \Lambda_k).\prod_{n=1}^{N} q(z_{nk})\right) \tag{3.49}$$

Dans un premier temps, nous allons appliquer la formule (3.41) pour chacun des termes de l'équation (3.49) ; cela nous donnera une expression pour chaque terme variationnel optimal, les autres estimateurs restant fixés. Nous déclinerons ensuite un algorithme à partir de ces expressions.

Termes variationnels optimaux

En appliquant la formule (3.41) pour un des termes $q(z_{nk})$, on obtient :

$$\ln q^*(z_{nk}) = z_{nk} \ln \rho_{nk} + \text{const} \tag{3.50}$$

$$\text{ou encore } q^*(z_{nk}) \propto \rho_{nk}^{z_{nk}} \tag{3.51}$$

avec :

$$\ln \rho_{nk} = \mathbb{E}[\ln \omega_k] + \frac{1}{2}\mathbb{E}[\ln \det \Lambda_k] - \frac{1}{2}\mathbb{E}[(\mathbf{x}_n - \mu_k)^T \Lambda_k (\mathbf{x}_n - \mu_k)] \tag{3.52}$$

Cette dernière expression fait intervenir des moments pris selon les autres termes variationnels courants $q^*(\theta_{/\mathbf{Z}})$, maintenus fixés à cette étape. Quand la forme fonctionnelle de ces derniers sera déterminée, nous donnerons les moments associés. Dans l'expression (3.51), nous pouvons reconnaitre la forme d'une distribution multinomiale. Avec la normalisation adéquate, nous pouvons donc calculer les moments de $q^*(\mathbf{Z})$:

$$\mathbb{E}[z_{nk}] = r_{nk} \tag{3.53}$$

$$\text{avec } r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}} \tag{3.54}$$

Par commodité pour la suite des développements, on définit quelques statistiques à partir de ces moments :

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{3.55}$$

$$\bar{\mathbf{x}}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{3.56}$$

$$\mathbf{S}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \bar{\mathbf{x}}_k)(\mathbf{x}_n - \bar{\mathbf{x}}_k)^T \tag{3.57}$$

Appliquons maintenant (3.41) à $q(\omega)$. On obtient :

$$\ln q^*(\omega) = \sum_{k=1}^{K} (\alpha_{0k} - 1) \ln \omega_k + \sum_{k=1}^{K} \sum_{n=1}^{N} r_{nk} \ln \omega_k + \text{const} \tag{3.58}$$

On reconnait la forme d'une distribution de Dirichlet. Nous avions vu précédemment que l'usage de distributions conjuguées amenait cette propriété. Cette distribution étant régulée par un hyper-paramètre, nous aurons donc un hyper-paramètre a posteriori $\alpha = \{\alpha_k\}$. Il suffira de mettre à jour celui-ci pour déterminer complètement le nouveau terme variationnel optimal. D'après l'expression (3.58), cette mise à jour s'effectue comme suit :

$$\alpha_k = \alpha_{0k} + N_k, \, \forall k \tag{3.59}$$

Connaissant maintenant la nature de $q^*(\omega)$, les moments selon cette distribution, requis par l'expression (3.52), peuvent être déterminés :

$$\mathbb{E}[\ln \omega_k] = \psi(\alpha_k) - \psi\left(\sum_k \alpha_k\right) \tag{3.60}$$

avec $\psi(.)$ la fonction digamma.

Il reste à appliquer (3.41) à $q(\mu, \Lambda)$. Ayant défini l'a priori $p(\mu_k, \Lambda_k)$ avec une distribution gaussienne-Wishart, le terme variationnel optimal doit avoir la même forme fonctionnelle. En effet, on trouve :

$$q^*(\mu_k, \Lambda_k) = \mathcal{N}(\mu_k | (\beta_k \Lambda_k)^{-1}) \mathcal{W}(\Lambda_k | \mathbf{W}_k, \nu_k) \tag{3.61}$$

Avec, de nouveau, une distribution caractérisée par des hyper-paramètres mis à jour comme suit :

$$\beta_k = \beta_{0k} + N_k \tag{3.62}$$

$$\mathbf{m}_k = \frac{1}{\beta_k} (\beta_{0k} \mathbf{m}_{0k} + N_k \bar{\mathbf{x}}_k) \tag{3.63}$$

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k \mathbf{S}_k + \frac{\beta_{0k} N_k}{\beta_{0k} + N_k} (\bar{\mathbf{x}}_k - \mathbf{m}_{0k})(\bar{\mathbf{x}}_k - \mathbf{m}_{0k})^T \tag{3.64}$$

$$\nu_k = \nu_{0k} + N_k \tag{3.65}$$

Les moments requis par l'expression (3.52) peuvent alors être extraits :

$$\mathbb{E}[(\mathbf{x}_n - \mu_k)^T \Lambda_k (\mathbf{x}_n - \mu_k)] = \frac{d}{\beta_k} + \nu_k (\mathbf{x}_n - \mathbf{m}_k)^T \mathbf{W}_k (\mathbf{x}_n - \mathbf{m}_k) \tag{3.66}$$

$$\mathbb{E}[\ln \det \Lambda_k] = \sum_{i=1}^{d} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + d \ln 2 + \ln \det \mathbf{W}_k \tag{3.67}$$

Algorithme

Utilisant les expressions et remarques déclinées précédemment dans cette section, nous proposons l'algorithme EM variationnel VBGMM (pour *Variational Bayesian Gaussian Mixture Model*) :

Entrées : Echantillon $\mathbf{X}$
Sorties : Modèle a posteriori θ'
Initialiser K avec une borne supérieure du nombre de composantes souhaitable et inconnu ;
Initialiser les hyper-paramètres variationnels d'un modèle θ à K composantes avec l'a priori (cf paragraphe «paramétrage») ;
Calcul des moments (3.66) et (3.67) ;
tant que *pas convergence* **faire**
    mettre à jour $q^*(\mathbf{Z})$ (eqn. 3.53) ;
    calculer les statistiques (3.55), (3.56) et (3.57) ;
    mettre à jour $q^*(\theta_{/\mathbf{Z}})$ (eqns. 3.62, 3.63, 3.64 et 3.65) ;
    Calcul des moments (3.66) et (3.67) ;
fin
Construire θ' avec les $K' < K$ composantes pour lesquelles $N_k \neq 0$ significativement ;

Algorithme 2 : Algorithme VBGMM d'estimation d'un mélange de gaussiennes

Nous complétons cet algorithme de quelques remarques, qui seront, dans le prochain paragraphe, illustrées par des expériences :

- Sans connaissance du nombre de groupes optimal, et avec d assez élevé, il est rapidement souhaitable d'initialiser K avec de grandes valeurs (i.e. la valeur de K nécessaire pour assurer une densité de l'initialisation suffisante dans tout le domaine des données suit e^d ; quand d devient élevé, c'est la *malédiction de la dimensionalité*). L'efficacité algorithmique d'EM variationnel sera donc pénalisée par la mise à jour de ce grand nombre de composantes. Cependant, avec l'a priori utilisé sur ω, on observe en général que le nombre de composantes «actives» (i.e. celles dont $N_k \neq 0$, ou de manière équivalente $\alpha_k \neq \alpha_0$) converge très rapidement ; il est donc envisageable de les détecter et de les supprimer à chaque itération, allégeant ainsi la charge de calcul au fil de l'avancée de l'algorithme.

– Nous présentons ici une approche gourmande (*greedy*) en allouant un grand nombre de composantes qui sera in fine réduit au strict nécessaire. Certains auteurs ont proposé des alternatives, notamment une stratégie de «naissance-et-mort» (*birth and death*) [Beal, 2003]. La mort d'une composante est indiquée assez simplement par le critère utilisé dans l'algorithme 2. La naissance exploite la structure de la borne inférieure (3.42) ; en effet, celle-ci peut être décomposée en termes de vraisemblance associés avec chacune des composantes. Parmi les composantes modélisant le moins bien les données, on en sélectionne alors une pour un «split», de manière analogue à la proposition de Ueda et al. [Ueda et al., 2000], constituant ainsi un nouveau modèle. Celui-ci est entrainé jusqu'à convergence ; si sa borne inférieure est augmentée relativement à l'ancien modèle, celui-ci est accepté.

– Les valeurs de borne inférieure dépendent de l'échelle et de la dispersion des données, rendant leur valeurs parfois difficiles à interpréter ; le critère de convergence est alors ambigu. Alternativement, Beal [Beal, 2003] propose une mesure d'agitation sur la distribution $q(\mathbf{Z})$ pour décider de la convergence. A l'itération t, celle-ci est définie par :

$$\text{agitation}(k)^{(t)} = \frac{\sum_{n=1}^{N} |q(z_{nk}=1)^{(t)} - q(z_{nk}=1)^{(t-1)}|}{N_k^{(t)}} \tag{3.68}$$

Où la notation $(.)^{(t)}$ indique une variable considérée à l'instant t. Ainsi, l'agitation moyenne sur les composantes «actives» (i.e. pour lesquelles $N_k \neq 0$) constitue un critère alternatif à la variation de borne inférieure. Au-delà d'une relative simplicité, celui-ci a l'avantage de ne plus dépendre de la taille et de la nature des données ou du modèle.

3.5.3 Comparaison expérimentale d'EM et VBGMM

Dans cette section nous présentons quelques expériences simples illustrant la nuance entre les algorithmes EM et variationnel pour estimer un mélange de gaussiennes. Après avoir rapidement présenté les données utilisées, nous définissons un protocole expérimental. Les résultats sont présentés et interprétés.

Données utilisées

Pour ces expériences, nous allons utiliser deux échantillons de données assez simples :

– **iris** [Fisher, 1936] : 150 individus décrits sur 4 attributs numériques, associés à des caractéristiques de fleurs (longueurs des pétales, etc). Ces éléments sont répartis en 3 classes réelles de 50 éléments. La figure 3.8 montre une projection de l'échantillon sur les 2 premiers attributs, en associant chaque individu à un motif spécifique à sa classe.
– **clust** : échantillon synthétique de 1000 individus obtenu selon 7 gaussiennes 2D bien séparées, avec des matrices de covariance aléatoires. Chaque classe comporte

environ $\frac{1000}{7}$ éléments. La classe de chaque élément est définie par la gaussienne l'ayant généré. Ces données sont représentées avec leur classe respective sur la figure 3.9.

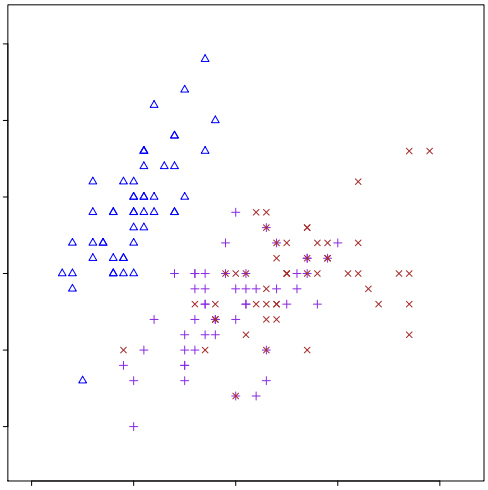


FIG. 3.8 – Echantillon *iris*

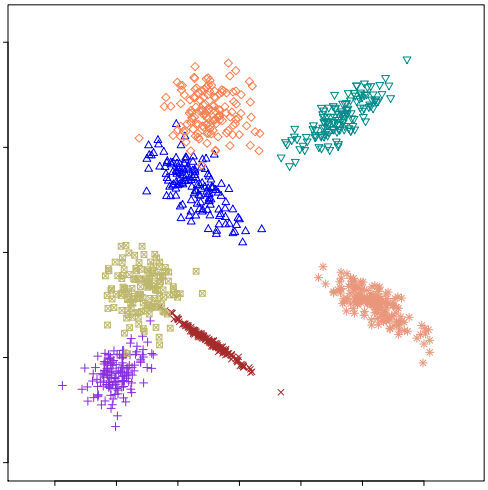


FIG. 3.9 – Echantillon *clust*

Protocole expérimental

Nous allons appliquer les algorithmes EM (i.e. l'algorithme 1) et VBGMM (i.e. l'algorithme 2) afin de réaliser des estimations de densité et des classifications de nos échantillons, sans tenir compte de la classe réelle des éléments (i.e. de manière non-supervisée).

Les classes réelles (ou étiquettes) de nos échantillons seront utilisées pour évaluer la qualité de nos estimations.

Nous souhaitons découvrir automatiquement le nombre de groupes adéquat pour chaque échantillon, et évaluer les capacités respectives d'EM et VBGMM à estimer de bons modèles. Pour cela, dans un premier temps nous appliquerons VBGMM avec $K = 100$, ce qui nous donnera un premier estimateur pour le nombre de groupes optimal pour chaque échantillon. L'expérience sera répétée 50 fois afin d'y associer une indication de confiance. Les caractéristiques suivantes seront relevées :

- l'évolution du nombre de composantes actives (i.e. celles pour lesquelles $N_k \neq 0$ pour les premières itérations de l'algorithme),
- l'évolution de l'agitation moyenne des composantes actives pour les premières itérations (voir formule 3.68),
- le nombre d'itérations avant convergence,
- le nombre de groupes finalement obtenu,

– la log-vraisemblance et le critère BIC [Schwarz, 1978] pour l'échantillon et le modèle considérés.

En partitionnant un échantillon de N éléments en K groupes, on infère un ensemble de N étiquettes (ou labels), chaque label prenant ses valeurs dans $\{1 \dots K\}$. Nous mesurons la qualité d'un ensemble de labels inférés (ou, de manière équivalente, son erreur vis-a-vis d'une vérité terrain ou de classes réelles) en comparant les labels inférés l_i aux vrais labels, ou classes T_i ; l'erreur n'augmentera que si $l_i \neq T_i$.

Toutefois, cette mesure, appliquée naïvement, peut être inconsistante ; le partitionnement est une opération non-supervisée, donc n'importe quelle permutation parmi les modalités d'étiquettes devrait être également valide. De plus, mesurer des inégalités strictes entre labels pose problème si le vrai nombre de classes K est différent du nombre de classes inféré K', un cas courant pour des problèmes réels.

Ces observations motivent l'utilisation d'une autre mesure d'erreur, suggérée dans [Fowlkes and Mallows, 1983, Azzag, 2005, Picarougne et al., 2007]. Celle-ci repose sur les valeurs d'étiquettes prises pour tous les couples possibles d'éléments de notre échantillon ; 2 éléments devraient avoir le même label inféré seulement si les vrais labels sont identiques, et réciproquement. Seules les violations de ces deux règles résulteront en une augmentation de l'erreur. Celle-ci est normalisée par $\frac{N(N-1)}{2}$, le nombre de couples distincts possibles. Cela nous garantit une erreur comprise entre 0 et 1.

Nous mesurons également la qualité de la densité de probabilité du modèle inféré, en calculant la divergence KL de celui-ci par rapport à une densité de référence. Pour l'échantillon *clust*, cette référence est simplement le modèle ayant généré nos données synthétiques. Pour *iris*, nous estimons une gaussienne sur les éléments de chaque classe réelle, pris séparément : la densité de référence est alors le mélange obtenu par le regroupement de ces 3 gaussiennes. Il n'existe pas d'expression analytique pour la divergence KL entre mélanges de gaussiennes, mais celle-ci peut être facilement estimée au moyen de simulations Monte-Carlo.

Nous verrons ensuite comment l'algorithme EM et le critère BIC peuvent permettre de découvrir ce nombre de groupes. L'algorithme EM sera exécuté pour plusieurs complexités de modèle différentes ($\{1, 2, 3, 4, 5\}$ pour *iris*, et $\{5, 7, 10, 12, 15\}$ pour *clust*), et une moyenne sur les valeurs de critère BIC obtenu sera établie sur 20 exécutions pour chaque paramétrage.

Pour la valeur minimisant ce critère, la log-vraisemblance, l'erreur de classification et la divergence KL par rapport à la densité de référence seront données. Ces résultats seront comparés à ceux obtenus avec VBGMM.

Résultats

	nb. itérations	K'	log-vraisemblance
iris	19.1 ± 8.8	3.7 ± 0.9	-207.8 ± 6.1
clust	42.9 ± 7.3	7.2 ± 0.3	-5423.0 ± 5.3
	BIC	erreur	div. KL
iris	695.3 ± 62.5	$10\% \pm 6\%$	0.23 ± 0.07
clust	11143.6 ± 20.1	$1\% \pm 0.2\%$	0.15 ± 0.08

TAB. 3.1 – Résultats de l'algorithme VBGMM

Dans la table 3.1, nous voyons que l'algorithme VBGMM permet d'estimer assez sûrement le nombre de composantes, si on prend en considération les vérités terrain de nos échantillons. L'erreur constatée est assez basse pour les deux échantillons.

Les courbes 3.10 et 3.12 montrent bien une décroissance rapide du nombre de composantes jouant un rôle dans le modèle ; une grande majorité de composantes ne joue plus aucun rôle après quelques itérations (plus que 10 composantes actives pour *iris* après 5 itérations, et 20 pour *clust* après 15 itérations). Cette observation illustre la viabilité de la variante suggérée à la fin de la section 3.5.2 : en initialisant VBGMM avec un K trop grand, le coût initial est élevé mais rapidement allégé au fur et à mesure des itérations.

Les courbes 3.11 et 3.13 montrent que l'agitation des affectations décroit quasi linéairement : cela concorde avec la vitesse de convergence linéaire évoquée précédemment [Celeux and Govaert, 1992].

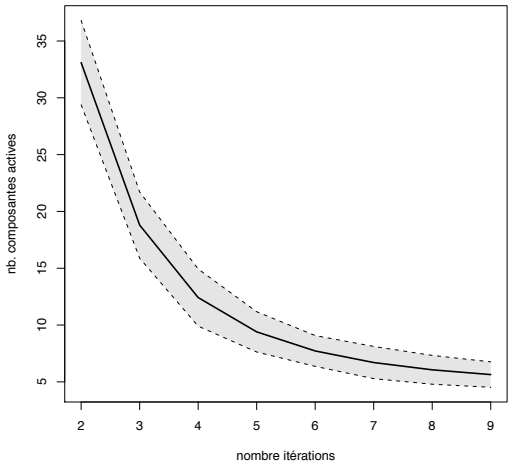


FIG. 3.10 – Evolution du nombre de composantes actives pour *iris*. La variabilité des résultats est illustrée en gris

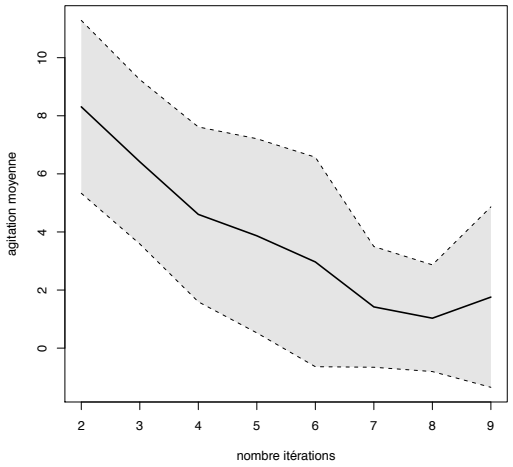


FIG. 3.11 – Evolution de l'agitation moyenne pour *iris*

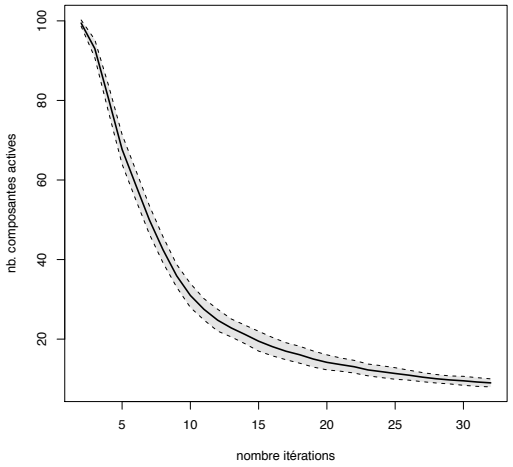


FIG. 3.12 – Evolution du nombre de composantes actives pour *clust*

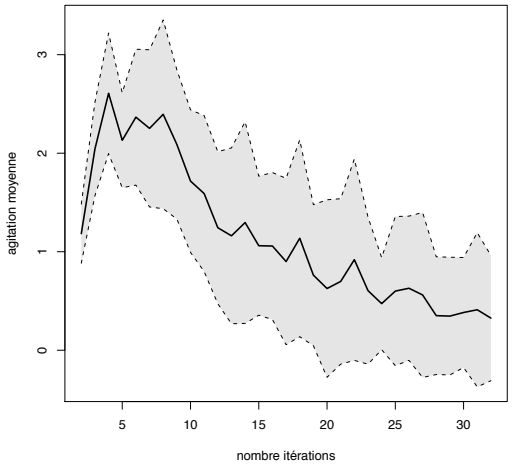


FIG. 3.13 – Evolution de l'agitation moyenne pour *clust*

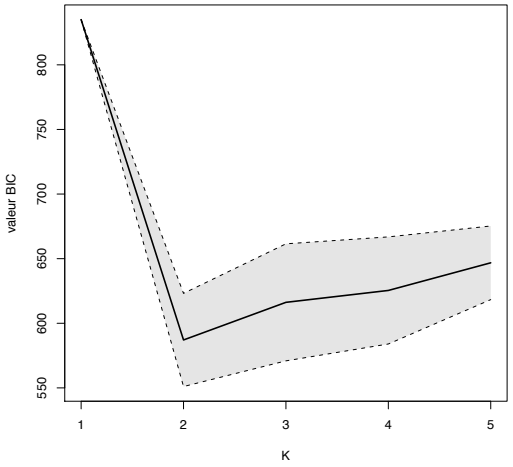


FIG. 3.14 – Critère BIC en fonction de la complexité du modèle pour *iris*

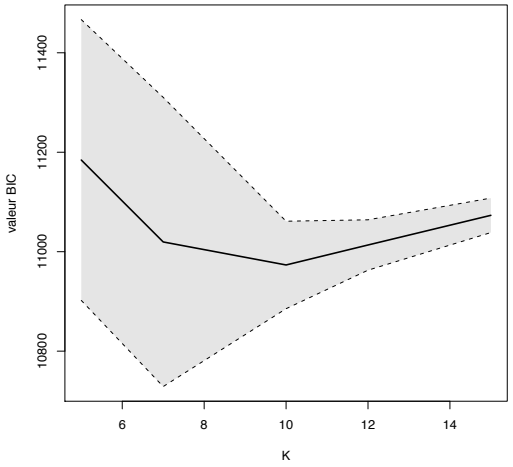


FIG. 3.15 – Critère BIC en fonction de la complexité du modèle pour *clust*

En utilisant l'algorithme EM classique, l'évaluation du critère BIC nous indique $K_{\text{iris}} = 2$ et $K_{\text{clust}} = 10$ (voir figures 3.14 et 3.15). Nous notons que les valeurs de BIC optimales obtenues avec EM sont souvent bien meilleures que pour VBGMM (voir table 3.1).

	nb. itérations	log-vraisemblance	erreur	div. KL
iris	12.6 ± 11.4	-218.4 ± 18.0	$22.9\% \pm 2.3\%$	0.25 ± 0.16
clust	33.2 ± 9.5	-5288.8 ± 44.6	$3.4\% \pm 2.4\%$	0.06 ± 0.04

TAB. 3.2 – Résultats de l'algorithme EM avec $K_{\text{iris}} = 2$ et $K_{\text{clust}} = 10$

La table 3.2 nous permet d'établir une comparaison entre EM et VBGMM, et de tirer les conclusions suivantes :

– Le nombre d'itérations avec EM est plus faible, mais compte tenu de la capacité de VBGMM à trouver le bon nombre de composantes à partir d'un K initial élevé sans passer par un critère de complexité, le coût impliqué par un seul processus de VBGMM est beaucoup plus faible que celui de l'approche classique.

– Le critère BIC est connu pour avoir tendance, sous certaines conditions, à pénaliser trop lourdement la complexité des modèles [McLachlan and D., 2000, chap. 6.9]. C'est le cas pour les modèles estimés sur les données *iris* : VBGMM trouve un nombre de composantes plus conforme à la réalité, pour une meilleure vraisemblance, une erreur plus faible, et une meilleure divergence KL, mais BIC est nettement meilleur pour $K = 2$.

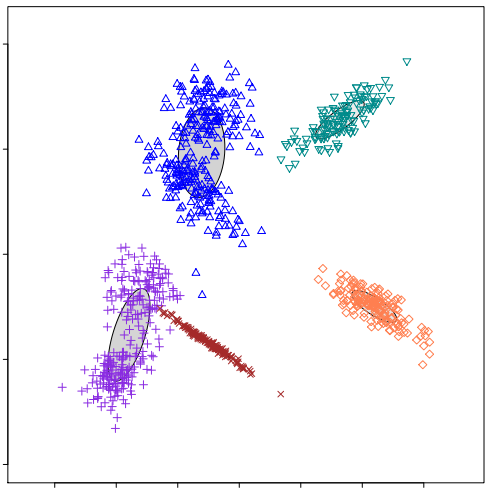


FIG. 3.16 – Plusieurs groupes modélisés par une composante avec EM et $K = 7$

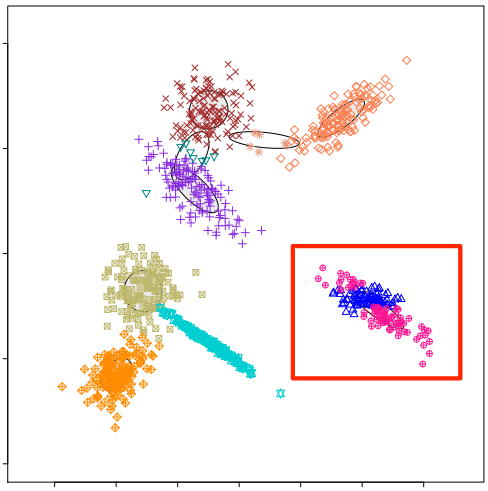


FIG. 3.17 – Composante centrale superflue sur un groupe avec EM et $K = 10$

– Par rapport à la vérité terrain de *clust*, un nombre trop grand de composantes est estimé avec EM et BIC. Quand on utilise $K = 7$, EM a tendance à estimer un modèle comme indiqué sur la figure 3.16 (les ellipses indiquent les courbes de niveau des composantes gaussiennes associées) : on se retrouve souvent avec une composante «à cheval» sur 2 groupes réels. La figure 3.17 montre un exemple du modèle typiquement obtenu pour $K = 10$: comme K est trop élevé par rapport aux nombre de groupes réels, deux composantes se retrouvent ajustées sur le même groupe (encadré en rouge dans la figure). La composante «centrale» est très concentrée, ce qui entraine des valeurs de vraisemblance très élevées ; toutefois, aucune dégénérescence algorithmique traitée par l'algorithme 1 n'est relevée. C'est un cas de mauvais maximum local [McLachlan and D., 2000] chap 3.10.

In fine nous voyons que VBGMM permet, en un seul processus de complexité acceptable, d'obtenir des modèles fiables. En réalisant une optimisation sur un espace de fonctions, régularisée par l'utilisation d'a priori peu informatifs, on évite les mauvais maxima locaux problématiques de l'approche EM classique.

3.5.4 Exemple du mélange d'ACP

Nous développons maintenant l'estimation variationnelle du mélange d'ACP probabilistes. Après avoir rappelé la définition du modèle, nous évoquerons rapidement le maximum de vraisemblance associé, puis présenterons en détail l'algorithme EM variationnel pour obtenir un tel maximum local respectant de bonnes propriétés. Nous proposons une discussion sur le paramétrage et les limites de cette approche, ayant fait

l'objet d'une contribution personnelle [Bruneau et al., 2010b,c].

L'ACP probabiliste

L'ACP (ou PCA pour *Principal Components Analysis*, dans la littérature anglophone) est une technique populaire et élémentaire pour la réduction de la dimensionalité de données numériques.

Considérant un échantillon de données $\mathbf{Y}$ défini sur $\mathbb{R}^d$, le sous-espace principal est généralement obtenu en diagonalisant la covariance de l'échantillon, i.e. en trouvant les vecteurs propres et les valeurs propres de cette matrice $d \times d$.

Tipping [Tipping and Bishop, 1999a] a proposé un cadre alternatif et probabiliste à l'ACP, fondé sur l'hypothèse que chaque élément de l'échantillon $\mathbf{y}$ est généré en transformant une variable aléatoire $\mathbf{x} \backsim \mathcal{N}(\mathbf{0}, \mathbf{I}_q)$, de dimensionalité $q < d$, et en ajoutant du bruit gaussien isotropique (i.e. covariance $= \sigma^2 \mathbf{I}_d$).

$$\mathbf{y} = \Lambda \mathbf{x} + \mu + \epsilon \tag{3.69}$$

Définissons maintenant les densités de probabilité associées :

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\Lambda\mathbf{x} + \mu, \sigma^2\mathbf{I}_d) \tag{3.70}$$
$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I}_q) \tag{3.71}$$
$$p(\epsilon) = \mathcal{N}(\epsilon|\mathbf{0}, \sigma^2\mathbf{I}_d) \tag{3.72}$$

Les propriétés des modèles linéaires gaussiens [Bishop, 2006, chap. 2], nous permettent de calculer la distribution marginale de $\mathbf{y}$:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mu, \Lambda\Lambda^T + \tau^{-1}\mathbf{I}_d) \tag{3.73}$$

avec $\tau = \sigma^{-2}$. Λ est une matrice $d \times q$, habituellement nommée matrice des facteurs. En d'autres termes, chaque élément $\mathbf{y}$ est construit par l'addition de μ, d'une combinaison linéaire des colonnes de Λ (colonnes désormais nommées *facteurs*), et de bruit isotropique. $\mathbf{x}$ contient les coefficients de la combinaison linéaire. Nous définissons également la notation compacte :

$$\mathbf{C} = \Lambda\Lambda^T + \tau^{-1}\mathbf{I}_d \tag{3.74}$$

Ce modèle sera nommé PPCA (*Probabilistic Principal Components Analysis*) par la suite. La figure 3.18 illustre la construction qu'effectue ce modèle.

La preuve a été faite que le maximum de vraisemblance de Λ est formé par les q vecteurs propres principaux de l'échantillon, mis à l'échelle par leurs valeurs propres respectives [Tipping and Bishop, 1999a]. Il n'existe cependant pas d'expression analytique pour un estimateur, qui ne peut être obtenu que par une procédure itérative. En

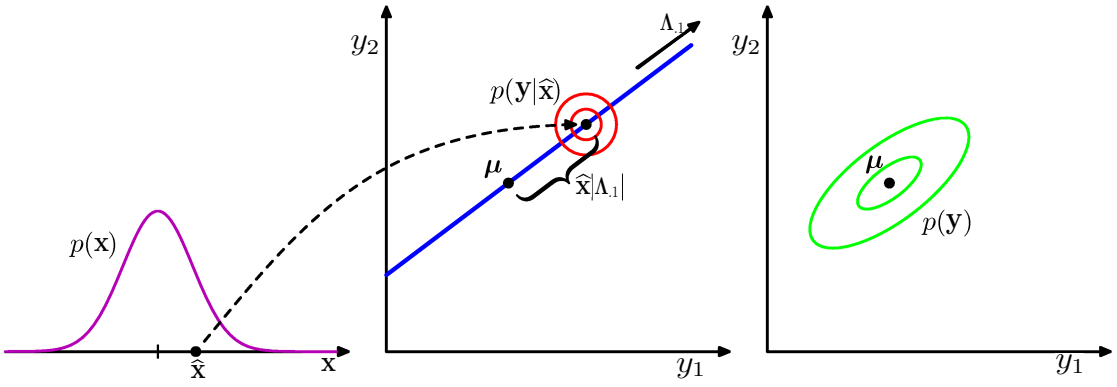


FIG. 3.18 – Illustration du modèle PPCA (Extrait de [Bishop, 2006]).

particulier, il est possible de trouver des formules de mise à jour pour $\theta = \{\mu, \Lambda, \tau\}$ et les variables latentes $\mathbf{x}$ en différenciant l'expression de vraisemblance. L'ensemble de formules de mise à jour obtenu implémente un algorithme EM [Tipping and Bishop, 1999a].

Il existe une classe d'équivalence d'estimateurs ML pour le modèle PPCA, modulo une matrice de rotation quelconque. Cependant, cette matrice peut-être récupérée en diagonalisant $\Lambda_{ML}^T \Lambda_{ML}$ [Tipping and Bishop, 1999a]. Ce calcul est d'un coût modéré, cette matrice étant $q \times q$. Ainsi, en post-multipliant la solution Λ_{ML} particulière obtenue par la matrice de rotation, on obtient en colonnes les vecteurs propres, mis à l'échelle selon leur valeurs propres, en ordre décroissant de grandeur des valeurs propres.

Estimation du maximum de vraisemblance d'un mélange d'ACP probabilistes (MPPCA)

Un modèle de mélange de PPCA est assez naturellement défini en introduisant une variable latente $\mathbf{z}$ indiquant l'appartenance d'un élément à un modèle PPCA particulier (nommé composante par la suite). De manière analogue au mélange de gaussiennes, un ensemble de poids $\{\omega_k\}$ est associé à K composantes pour décrire leur importance relative, et une variable latente $\mathbf{z}$ *1-parmi-K* (voir formule (3.16)).

Ainsi, une densité multimodale est estimée sur l'échantillon de données, et chaque composante détermine son propre sous-espace principal. Les densités associées sont :

$$p(\mathbf{z}|\theta) = \prod_k^K \omega_k{}^{z_k} \quad p(\mathbf{y}|\mathbf{z},\theta) = \prod_k^K \mathcal{N}(\mathbf{y}|\mu_k, \mathbf{C}_k)^{z_k} \tag{3.75}$$

$$p(\mathbf{y}|\theta) = \sum_k^K \omega_k \mathcal{N}(\mathbf{y}|\mu_k, \mathbf{C}_k) \tag{3.76}$$

En conséquence, un échantillon de données $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ a pour fonction de vraisemblance :

$$p(\mathbf{Y}|\theta) = \prod_n^N \sum_k^K \omega_k \mathcal{N}(\mathbf{y}_n|\mu_k, \mathbf{C}_k) \tag{3.77}$$

Les variables latentes associées à notre échantillon sont collectivement dénotées par $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ et $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. Dans l'expression (3.77), $\mathbf{X}$ et $\mathbf{Z}$ sont implicites. En développant cette expression avec (3.75), (3.71) et (3.70), les variables latentes apparaissent désormais, impliquant notamment une dépendance entre $\mathbf{X}$ et $\mathbf{Z}$.

L'expression de vraisemblance ainsi définie peut être différenciée selon chaque paramètre et variable latente, de manière analogue au cas à une composante : un algorithme EM peut donc de nouveau être constitué. L'estimateur ML local obtenu nous fournit un modèle a posteriori pour l'échantillon considéré. Toutefois, à l'instar de l'algorithme EM pour le mélange de gaussiennes, le nombre de composantes à estimer K et le nombre de facteurs de chaque composante q doivent être connus à l'avance.

Dans un contexte d'apprentissage non-supervisé cela est rarement le cas, aussi ce problème est traditionnellement réglé en utilisant des critères de sélection de modèles évoqués dans la section 3.4.5. Des modèles avec plusieurs valeurs de K et q sont estimés, et ces critères aident à choisir quelles complexités devraient être retenues. Cette approche souffre cependant des mêmes limitations que celles précisées dans la section 3.4.4.

Bishop a décrit un algorithme EM variationnel pour l'estimation d'une seule composante PPCA [Bishop, 1999], et Beal et Ghahramani ont traité de la même manière le mélange d'analyseurs factoriels (FA) [Beal, 2003, Ghahramani and Beal, 2000].

FA et PPCA diffèrent seulement par le modèle de bruit utilisé ; d'après l'expression (3.72), il est défini isotropique pour PPCA, alors qu'il est choisi diagonal pour FA. Par conséquent, l'invariance rotationnelle mentionnée dans la section 3.5.4 n'est plus valable pour FA, remplacée par une invariance vis-à-vis de la mise à l'échelle de chaque dimension de l'échantillon [Tipping and Bishop, 1999a].

L'apprentissage variationnel de mélanges de distributions est généralisé à toute la famille exponentielle dans [Watanabe et al., 2009]. Toutefois, les auteurs se différencient de notre formalisme du MPPCA en ce qu'ils réalisent l'apprentissage d'un sous-espace commun à toutes les composantes du modèle. Les travaux de Beal [Beal, 2003] ont été adaptés par nos soins au cas du MPPCA, augmentés de quelques considérations algorithmiques [Bruneau et al., 2010b,c]. Nous résumons ces travaux dans la suite de la section.

Algorithme variationnel pour l'estimation du MPPCA

Les propriétés de l'approche variationnelle permettent d'envisager le règlement des problèmes inhérents à l'approche ML présentée précédemment.

Il faut pour cela, dans un premier temps, choisir un a priori adapté, pour des composantes bien séparées dans un modèle à complexité adéquate. L'application du théorème 3.5.1 permet alors de décliner un algorithme EM variationnel, garantissant un maximum local d'une fonction objectif qui respecte les propriétés spécifiées dans [Xiang and Gong, 2006].

Nous définissons les distributions a priori comme suit :

$$p(\omega|\alpha_0) = \prod_k^K p(\omega_k|\alpha_0) = \prod_k^K \mathrm{Dir}(\omega_k|\alpha_{0k}) \tag{3.78}$$

$$p(\Lambda|\nu) = \prod_k^K p(\Lambda_k|\nu_k) \tag{3.79}$$

$$= \prod_k^K \prod_j^q p(\Lambda_k^{\cdot j}|\nu_{kj}) = \prod_k^K \prod_j^q \mathcal{N}(\Lambda_k^{\cdot j}|\mathbf{0}, \nu_{kj}^{-1}\mathbf{I}_d) \tag{3.80}$$

$$p(\nu|a_0, b_0) = \prod_k^K p(\nu_k|a_0, b_0) = \prod_k^K \prod_j^q \mathrm{Ga}(\nu_{kj}|a_0, b_0) \tag{3.81}$$

$$p(\mu|\mu_0, \nu_0) = \prod_k^K p(\mu_k|\mu_{0k}, \nu_0) = \prod_k^K \mathcal{N}(\mu_k|\mu_{0k}, \nu_0^{-1}\mathbf{I}_d) \tag{3.82}$$

Avec ν_{kj} les paramètres de précision des facteurs, et $\mathrm{Ga}(.|a, b)$ la distribution Gamma paramétrée par a et b, respectivement l'échelle et la forme inverse de la distribution.

En complétant le modèle du MPPCA par cet a priori, on peut vérifier, sur la figure 3.19, que nous définissons alors un DAG. Ceci confirme que l'implémentation de l'algorithme EM variationnel existe pour ce modèle.

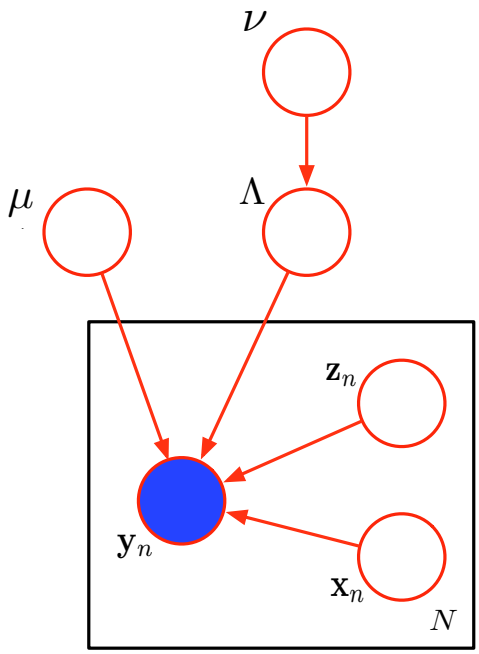


FIG. 3.19 – Représentation du mélange d'ACP par un DAG

En pratique, ces a priori vont être instanciés de manière à favoriser certaines propriétés :

- on choisit $\alpha_{0k} = 10^{-3}\ \forall k$ de manière à favoriser le moins de composantes possibles supportant des données,
- a_0 et b_0 sont fixés à 10^{-3} pour assurer qu'aucun facteur n'ait un rôle prépondérant a priori,
- Les μ_{0k} sont choisis aléatoirement et uniformément dans le domaine de l'échantillon. ν_0 est fixé à 10^{-3}. Ceci permet une bonne exploration de l'espace des données, tout en ne biaisant pas significativement chaque composante.

Nous n'avons pas défini de distribution a priori pour τ (e.g. voir eqn. (3.73)). Cette valeur reflète les plus petites valeurs propres de la covariance de l'échantillon [Tipping and Bishop, 1999a]. Cette information est en général stable, nous avons donc choisi de ne pas l'optimiser, la gardant en simple paramètre. Par ailleurs, l'optimiser impliquerait une charge de calcul significative, et les résultats expérimentaux sont en général plus stables et précis quand τ est fixé à une valeur arbitraire en accord avec les autres hyper-paramètres a priori, 1 dans notre cas.

Comme précédemment, on résume collectivement les paramètres du modèle à estimer avec $\theta = \{\mu, \Lambda, \omega, \nu\}$.

Nous avons donc défini la vraisemblance complète, qui, en combinant les expressions (3.75), (3.71), (3.70), (3.78), (3.80), (3.81) et (3.82), peut être résumée comme suit :

$$\begin{aligned} p(\mathbf{Y}, \mathbf{X}, \mathbf{Z}, \theta | \alpha_0, a_0, b_0, \mu_0, \nu_0) = p(\mathbf{Z}|\omega) p(\mathbf{Y}|\mathbf{Z}, \mathbf{X}, \Lambda, \mu) p(\mathbf{X}|\mathbf{Z}). \\ p(\omega|\alpha_0) p(\Lambda|\nu) p(\nu|a_0, b_0) p(\mu|\nu_0) \end{aligned} \tag{3.83}$$

Suivant cette factorisation imposée par le modèle, nous définissons la distribution variationnelle associée :

$$\begin{aligned} q(\mathbf{X}, \mathbf{Z}, \theta) &= q(\mathbf{X}, \mathbf{Z}) q(\theta) \\ q(\mathbf{X}, \mathbf{Z}) &= \prod_n q(\mathbf{x}_n|\mathbf{z}_n) q(\mathbf{z}_n) \\ q(\theta) &= q(\omega) \prod_k^K q(\mu_k) q(\Lambda_k|\nu_k) q(\nu_k) \end{aligned} \tag{3.84}$$

On peut alors utiliser la distribution variationnelle (3.84) et la formule de vraisemblance complète (3.83) pour décliner un algorithme EM variationnel. Celui-ci nous permettra d'inférer la distribution $q^*(\mathbf{X}, \mathbf{Z}, \theta)$ optimale. La borne inférieure optimisée par notre algorithme peut être trouvée en appliquant la formule (3.42). Dans le cas présent, celle-ci est exprimée de la manière suivante :

$$\begin{aligned} \mathcal{F} &= \mathbb{E}\big[\ln p(\mathbf{Y}, \mathbf{X}, \mathbf{Z}, \theta | \alpha_0, a_0, b_0, \mu_0, \nu_0)\big] - \mathbb{E}\big[\ln q(\mathbf{X}, \mathbf{Z}, \theta)\big] \\ &= \mathbb{E}\big[\ln p(\omega|\alpha_0)\big] + H\big[q(\omega)\big] + \sum_{k=1}^{K} \Bigg(\mathbb{E}\big[\ln p(\nu_k|a_0, b_0)\big] + H\big[q(\nu_k)\big] + \mathbb{E}\big[\ln p(\Lambda_k|\nu_k)\big] \\ &\quad + H\big[q(\Lambda_k)\big] + \mathbb{E}\big[\ln p(\mu_k)\big] + H\big[q(\mu_k)\big] + \sum_{n=1}^{N} r_{nk} \Big(-\ln r_{nk} + \mathbb{E}[\ln \omega_k] \\ &\quad + \mathbb{E}\big[\ln p(\mathbf{x}_n|z_{nk}=1)\big] + H\big[q(\mathbf{x}_n|z_{nk}=1)\big] + \mathbb{E}\big[\ln p(\mathbf{y}_n|z_{nk}=1, \mathbf{x}_n, \Lambda_k, \mu_k)\big] \Big) \Bigg) \end{aligned} \tag{3.85}$$

où, conformément à la définition (3.42), les espérances sont prises selon les termes variationnels concernés, et $H[q]$ dénote l'entropie de la distribution q. De même, nous avons réutilisé la notation $\mathbb{E}[z_{nk}] = r_{nk}$, introduite dans le cadre de l'algorithme VBGMM.

L'expression (3.85) peut être retrouvée en utilisant les distributions variationnelles pour appliquer l'inégalité de Jensen sur la vraisemblance marginale $\mathcal{L}$. Les propriétés des distributions de probabilité facilitent les intégrations, et permettent de retrouver notre expression compacte.

Comme nous avons démontré dans la section 3.5.1, maximiser $\mathcal{F}$ revient à minimiser $\mathrm{KL}[q(\theta)||p(\theta|\mathbf{Y})]$. Il suffit d'appliquer la formule (3.41) pour chaque paramètre, définissant alors un ensemble de formules de mises à jour :

– **Etape E**

$$\begin{aligned}
&\Sigma_{\mathbf{x}_k} = \left(I_q + \tau\mathbb{E}[\Lambda_k^T\Lambda_k]\right)^{-1}\\
&\mathbb{E}_{q(\mathbf{x}_n|z_{nk}=1)}[\mathbf{x}_n] = \mathbb{E}[\mathbf{x}_{nk}] = \tau\Sigma_{\mathbf{x}_k}\mathbb{E}[\Lambda_k^T](\mathbf{y}_n - \mathbb{E}[\mu_k])\\
&\mathbb{E}_{q(\mathbf{x}_n|z_{nk}=1)}[\mathbf{x}_n\mathbf{x}_n^T] = \mathbb{E}[\mathbf{x}_{nk}\mathbf{x}_{nk}^T] = \Sigma_{\mathbf{x}_k} + \mathbb{E}[\mathbf{x}_{nk}]\mathbb{E}[\mathbf{x}_{nk}]^T\\
&r_{nk} \propto \exp\left(\psi(\alpha_k) - \psi\left(\sum_j \alpha_j\right) - \frac{1}{2}\mathrm{Tr}(\mathbb{E}[\mathbf{x}_{nk}\mathbf{x}_{nk}^T]) - \frac{\tau}{2}\left[\|\mathbf{y}_n\|^2 + \mathbb{E}[\|\mu_k\|^2]\right.\right.\\
&\left.\left. - 2(\mathbf{y}_n - \mathbb{E}[\mu_k])^T\mathbb{E}[\Lambda_k]\mathbb{E}[\mathbf{x}_{nk}] - 2\mathbf{y}_n^T\mathbb{E}[\mu_k] + \mathrm{Tr}(\mathbb{E}[\Lambda_k^T\Lambda_k]\mathbb{E}[\mathbf{x}_{nk}\mathbf{x}_{nk}^T])\right]\right)
\end{aligned} \tag{3.86}$$

– **Statistiques synthétiques**

$$\begin{aligned}
N_k &= \sum_n r_{nk} & \mathbf{y}_k^2 &= \sum_n r_{nk}||\mathbf{y}_n||^2\\
\mathbf{y}_k &= \sum_n r_{nk}\mathbf{y}_n & \mathbf{s}_k &= \sum_n r_{nk}\mathbb{E}[\mathbf{x}_{nk}]\\
\mathbf{sy}_k &= \sum_n r_{nk}\mathbf{y}_n\mathbb{E}[\mathbf{x}_{nk}]^T & \mathbf{S}_k &= \sum_n r_{nk}\mathbb{E}[\mathbf{x}_{nk}\mathbf{x}_{nk}^T]
\end{aligned} \tag{3.87}$$

– **Etape M**

$$\begin{aligned}
&\alpha_k = \alpha_{0k} + N_k && a_{kj} = a_0 + \frac{d}{2}\\
&b_{kj} = b_0 + \frac{1}{2}\sum_i^d \mathbb{E}[{\Lambda_k^{ij}}^2] && \Sigma_{\Lambda_k} = (\mathrm{diag}(\mathbb{E}[\nu_k]) + \tau\mathbf{S}_k)^{-1}\\
&\mathbb{E}[\Lambda_k^{i.}] = \Sigma_{\Lambda_k}\tau\left[\mathbf{sy}_k^{i.} - \mathbb{E}[\mu_k^i]\mathbf{s}_k\right] && \mathbb{E}[\Lambda_k^T\Lambda_k] = \sum_i^d \mathbb{E}[\Lambda_k^{i.}{\Lambda_k^{i.}}^T]\\
&\mathbb{E}[{\Lambda_k^{ij}}^2] = \mathbb{E}[\Lambda_k^{i.}{\Lambda_k^{i.}}^T]^{jj} && \Sigma_{\mu_k} = \left[\nu_0 + \tau N_k\right]^{-1} I_d\\
&\mathbb{E}[\mu_k] = \Sigma_{\mu_k}\left[\nu_0\mu_{0k} + \tau(\mathbf{y}_k - \mathbb{E}[\Lambda_k]\mathbf{s}_k)\right]
\end{aligned} \tag{3.88}$$

Ces expressions de mise à jour permettent de définir l'algorithme EM variationnel pour l'estimation d'une approximation au véritable a posteriori. Nous rappelons que, par analogie avec l'algorithme EM classique, les étapes E et M désignent respectivement

les mises à jour des a posteriori variationnels sur les variables latentes et les paramètres. Par la suite, nous nommerons cette instance d'algorithme, dédiée au modèle MPPCA, VBMPPCA.

La distribution variationnelle produite par cet algorithme sera une approximation de la distribution a posteriori réelle, mais inconnue, pour cet échantillon. D'après les propriétés générales des algorithmes variationnels, chaque itération sur les expressions de mise à jour augmentera $\mathcal{F}$, la convergence est donc facilement évaluée en surveillant cette valeur, et en arrêtant les opérations lorsque la variation de celle-ci entre deux itérations passe sous un certain seuil.

Nous rappelons que, quand l'algorithme EM classique optimise selon une valeur, les algorithmes variationnels optimisent dans l'espace des fonctions. Par ailleurs, l'utilisation de distributions de la famille exponentielle implique que les a posteriori variationnels ont la même forme fonctionnelle que leurs a priori respectifs. Ce sont des propriétés générales des algorithmes variationnels appliqués aux distributions de cette famille ([Attias, 2000], et aussi voir section 3.5.2).

Paramétrage

Nous avons déjà évoqué la question du paramétrage a priori dans la précédente section ; les choix évoqués peuvent être utilisés pour l'initialisation des paramètres variationnels. En effet, pour une distribution variationnelle à q facteurs, $q-1$ d'entre ceux-ci doivent d'abord être initialisés avant de pouvoir démarrer l'algorithme.

L'algorithme 3 reflète nos observations. Remarquons que le modèle MPPCA utilise le même a priori que l'algorithme 2 sur les poids : aussi, conformément aux remarques faites précédemment, nous ajoutons une étape de filtrage des composantes ne jouant plus aucun rôle, i.e. pour lesquelles $\alpha_k \simeq \alpha_{0k}$. Les facteurs sont initialisés avec des matrices orthogonales aléatoires, afin de refléter le fait que les espaces principaux doivent être orthogonaux.

L'a priori introduit par les expressions (3.80) et (3.81) implémente l'idée d'*Automatic Relevance Determination* (ou ARD dans la littérature [MacKay, 1995], soit détermination automatique de la pertinence) ; un paramètre de précision est associé à chaque colonne de la matrice de facteurs, et quand une colonne est amenée à ne jouer aucun rôle (i.e. les données que la composante supporte sont déjà bien modélisées par un ensemble de facteurs complémentaires), le mode de la distribution variationnelle associée prendra une grande valeur.

Les facteurs jouant un rôle effectif seront en revanche associés à un ν_{kj} faible. Ainsi, notre modélisation favorise les sous-espaces principaux de moindre complexité, et nous

Entrées : Un échantillon, une borne supérieure K au nombre de composantes effectif
Sorties : L'ensemble $\{\theta'\}$ des composantes effectives
Définir $\{\theta\}$ un ensemble de K composantes a priori ;
tant que *non convergence* **faire**
  Mettre à jour les variables latentes $\mathbf{X}$ et $\mathbf{Z}$ (Etape E) ;
  Mettre à jour les statistiques synthétiques des variables latentes ;
  Mettre à jour $\{\theta\}$ (Etape M) ;
fin
$\{\theta'\} \leftarrow$ ensemble vide ;
pour tous les *chaque composante* θ_k *dans* $\{\theta\}$ **faire**
  si $\alpha_k > \alpha_{0k}$ **alors**
    ajouter θ_k à $\{\theta'\}$;
  fin
fin

Algorithme 3 : Algorithme VBMPPCA estimant le nombre de composantes adéquat

fournit de plus un moyen de sélectionner automatiquement la dimensionalité sous-jacente à chaque composante.

Néanmoins, de manière analogue au dilemme du choix initial de K, une valeur initiale doit être choisie pour q ; la seule contrainte est $q < d$, aussi le choix naïf serait de fixer $q = d - 1$. Remarquons cependant que la complexité des équations de mise à jour suit q^2 ; aussi pour d très élevé ce choix initial est peu recommandable.

Après exécution de l'algorithme, chaque matrice de facteurs peut être ré-arrangée en la post-multipliant par une matrice de rotation (voir notre présentation du modèle PPCA). Ainsi, pour une composante donnée, au-delà d'un index de colonne q', le paramètre ARD associé dépassera un seuil, ainsi que tous ceux d'index supérieur. Ceci nous permet de définir l'a posteriori de Λ comme la matrice $d \times q'$ dont les colonnes sont associées à une valeur de précision faible.

Ce seuil étant dépendant de l'échelle et la distribution des données, il peut donc être souhaitable d'employer une technique plus sophistiquée. Nous exploitons la possibilité de récupérer les colonnes des matrices de facteurs dans l'ordre de ses valeurs propres respectives. En effet, nous pouvons alors choisir le nombre de facteurs q' à retenir en sélectionnant les q' premières colonnes de la matrice de facteurs. Ceux-ci sont utilisés pour la reconstruction de la matrice de covariance équivalente grâce à la formule (3.74). Nous proposons un algorithme simple, consistant à utiliser de 1 à q facteurs pour la reconstruction, en estimant q' comme le seuil à partir duquel la covariance équivalente

ne change plus à l'ajout de facteurs supplémentaires.

A chaque facteur ajouté, nous mesurons la variation de divergence KL ou Jensen-Shannon (JS). La dimensionalité locale adéquate peut alors être trouvée lorsque cette variation passe sous un seuil. Comme les mesures de divergence sont indépendantes de l'échelle des données, ce seuil peut être utilisé dans n'importe quelle situation.

Problèmes de biais et de convergence

Dans l'algorithme EM variationnel, chaque mise à jour de paramètres variationnels repose sur des moments évalués selon les autres facteurs de la distribution variationnelle. Dans le cas du mélange de gaussiennes (voir [Attias, 2000] et [Bishop, 2006, chap. 10]), chaque expression de mise à jour de l'étape M repose exclusivement sur les statistiques synthétiques extraites des variables latentes et de l'échantillon. Les conséquences de cette propriété sont :

- N'importe quel ordre peut être choisi pour les mises à jour au sein de l'étape M ; il faut seulement que chacune soit effectuée une fois,
- En d'autres termes, les moments utilisés n'évoluent pas au cours d'une itération.

L'inspection des expressions de mise à jour de VBMPPCA (expressions (3.86), (3.87) et (3.88)) nous amène aux conclusions suivantes :

- Au sein d'une étape E, nous remarquons que les variables latentes $\mathbf{X}$ ne dépendent pas de $\mathbf{Z}$, alors que l'inverse est vrai. Donc la mise à jour de $\mathbf{X}$ puis $\mathbf{Z}$ nous permet de conserver des estimateurs consistants. Ceux-ci sont utilisés pour calculer les statistiques synthétiques qui résument l'influence relative des variables latentes et de l'échantillon pour l'étape M à venir.

- L'étape M, en revanche, pose problème. Nous pouvons en effet y observer des couples de paramètres inter-dépendants (voir fig. 3.20). Cela signifie qu'aucune séquence naturelle ne peut être envisagée ; par exemple, si dans un premier temps μ est mis à jour, puis Λ, l'étape E qui suivra utilisera un estimateur pour μ a priori pas consistant, car le moment de sa distribution a été implicitement changé par la mise à jour de la distribution de Λ.

D'un point de vue statistique, des biais sont induits. Etant donné l'absence de séquence naturelle évoquée ci-dessus, aucune procédure systématique pour supprimer ce biais ne peut être envisagée. Les propriétés de l'algorithme variationnel sont toujours valables (i.e. $\mathcal{F}$ est strictement monotone croissante et bornée), mais dans certains cas, l'a posteriori local qui sera calculé peut être assez mauvais.

Ce problème a déjà été brièvement présenté par Beal [Beal, 2003], suggérant que des mises à jour répétées de $\mathbf{Z}$, μ, $\mathbf{X}$ et Λ, avant de mettre à jour ω et ν, au sein d'une

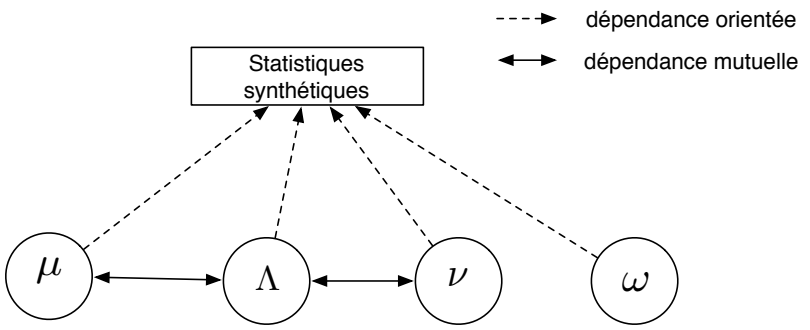


FIG. 3.20 – Inter-dépendances des moments des paramètres variationnels. Les variables latentes sont résumées collectivement par *statistiques synthétiques*.

même itération, peuvent augmenter plus efficacement $\mathcal{F}$. Cette adaptation peut aussi être vue comme un moyen empirique de stabiliser les estimateurs au cours de chaque itération.

S'il semble expérimentalement efficace, le choix effectué dans [Beal, 2003] casse la logique EM habituelle (i.e. des mises à jour de variables latentes et de paramètres sont entrelacées). Par ailleurs, dans notre analyse nous avons conclu qu'un biais était engendré uniquement par l'inter-dépendance des mises à jour de paramètres de l'étape M, les estimateurs des variables latentes n'étant pas en cause.

Dans l'algorithme 4, nous avons donc choisi de procéder à des mises à jour répétées des seuls μ et Λ au sein de chaque étape M (entre 5 et 10 fois semble un bon compromis), ces paramètres étant les plus fortement inter-dépendants. Relativement à la proposition de Beal, la notre présente l'avantage de conserver le schéma EM habituel.

Remarquons également qu'en utilisant des statistiques synthétiques, seule l'étape E présente une complexité dépendant de la taille des données : ceci signifie que le surplus calculatoire impliqué par nos mises à jour répétitives (voir ligne 6), circonscrit dans l'étape M, n'est pas dépendant d'une taille d'échantillon. Ce dernier point est important dans une optique de passage à l'échelle.

Les dépendances observées parmi les a posteriori des paramètres viennent de la simplicité relative du formalisme choisi, e.g. l'hypothèse d'indépendance entre μ et (Λ, ν). Dans la littérature, ce choix est motivé par l'introduction des a priori ARD pour la réduction du sous-espace de chaque composante. Il serait possible, en théorie, d'envisager un modèle avec un moindre degré de factorisation. Cependant, les calculs impliqués seraient alors nettement plus compliqués qu'ils ne le sont déjà ; ce constat a

Entrées : Un échantillon, une borne supérieure K au nombre de composantes effectives
Sorties : L'ensemble $\{\theta'\}$ des composantes effectives
Définir $\{\theta\}$ l'ensemble de K composantes a priori ;
tant que *non convergence* **faire**
  Mettre à jour $\mathbf{X}$ et $\mathbf{Z}$ (Etape E) ;
  Mettre à jour statistiques synthétiques ;
  Mettre à jour α ;
  Mettre à jour μ et Λ 5-10 fois ;
  Mettre à jour ν ;
fin
$\{\theta'\} \leftarrow$ ensemble vide ;
pour tous les *chaque composante* θ_k *dans* $\{\theta\}$ **faire**
  si $\alpha_k > \alpha_{0k}$ **alors**
    ajouter θ_k à $\{\theta'\}$;
  fin
fin

Algorithme 4 : Adaptation empirique de VBMPPCA

conduit au choix de la solution la plus simple.

Propriétés relatives de VBGMM et VBMPPCA

VBGMM est déjà capable de traiter un échantillon comprenant des données fortement corrélées, aussi en utilisant $q = d - 1$ on obtient généralement des solutions similaires pour VBGMM et VBMPPCA. La complexité algorithmique est également similaire pour les 2 algorithmes (c-a-d asymptotiquement linéaire selon la taille de l'échantillon), mais chaque itération de VBMPPCA est alourdie de la mise à jour des variables latentes $\mathbf{X}$.

Aussi, en tenant compte des problèmes d'initialisation et de convergence mis en lumière ci-dessus, il peut parfois être avantageux d'estimer un mélange de gaussiennes, et d'en effectuer la diagonalisation en post-traitement.

Toutefois, pour des données à très haute dimensionalité, pour lesquelles les rangs des sous-espaces locaux à chaque composante seront vraisemblablement bien inférieurs à d, nous pouvons initialiser q à une borne supérieure de ces rangs attendus, ainsi $q < d-1$. Le gain calculatoire dans chaque itération est quadratique selon la différence entre q et $d - 1$. Quand notre hypothèse de rangs locaux faibles tient approximativement, nous pouvons ainsi obtenir des modèles valides avec VBMPPCA mettant en oeuvre moins de ressources que VBGMM.

3.5.5 Variantes, extensions et alternatives

Algorithme variationnel incrémental

Smidl et Quinn [Smidl and Quinn, 2006] proposent une revue de l'adaptation de l'apprentissage variationnel au contexte d'un flux de données. Nous renvoyons le lecteur à la section 2.5 pour la nuance entre un algorithme incrémental (i.e. qui traite un sous-ensemble de l'échantillon à chaque itération), et un algorithme en ligne, traitant un flux de données dans l'ordre de son arrivée, avec une mémoire limitée. Nous utilisons l'abus de langage consistant à associer le caractère incrémental au traitement des données les plus récentes.

Supposons que le flux produit un élément de données tous les Δt. Dans cette section nous indexons les éléments de données avec t, au lieu de n habituellement : l'élément courant est alors $\mathbf{x}_t$, et les données reçues jusqu'à l'horizon t sont collectivement dénotées par $\mathbf{X}_t = \{\mathbf{x}_1, \dots \mathbf{x}_t\}$.

Dans [Smidl and Quinn, 2006], 2 modélisations distinctes sont envisagées :

– Les paramètres du modèle θ sont invariants dans le temps. Dans ce cas, le problème revient à réaliser une estimation récursive de θ tous les Δt. La fonction de vraisemblance utilisée dans le cadre de l'algorithme variationnel est alors :

$$p(\mathbf{x}_t|\theta, \mathbf{X}_{t-1}) \tag{3.89}$$

A chaque étape, on dispose d'un estimateur variationnel $\tilde{p}(\theta|\mathbf{X}_{t-1})$ (initialisé avec un a priori à $t = 0$), et l'objectif est d'approximer le modèle a posteriori $\tilde{p}(\theta|\mathbf{X}_t)$. Le théorème 3.5.1 peut alors être appliqué au cas incrémental sans problème pour toutes les classes de modèles θ où l'estimation hors-ligne était possible.

Pour les modèles de mélanges, le procédé repose sur des accumulateurs (ou statistiques synthétiques) ; dans le cas incrémental, on n'accède aux données qu'une fois, les affectations sont donc définitives. Cela signifie qu'au début de l'estimation, quand le modèle est encore très instable, un biais important peut être induit. Il faut alors s'assurer d'avoir un bon premier estimateur, soit avec un a priori judicieux, soit avec une phase d'initialisation hors-ligne.

D'autre part, il est a priori nécessaire d'itérer l'algorithme variationnel jusqu'à convergence tous les Δt : cela peut être problématique dans un contexte d'application en temps réel. Il est cependant possible de n'effectuer qu'une seule itération de l'algorithme variationnel à chaque arrivée d'un élément, la consistance des estimateurs alors obtenus ayant été démontrée [Sato, 2001]. Les modèles auto-régressifs (AR) ou mélanges d'AR sont les formes fonctionnelles couramment estimées de cette manière. Le mélange de gaussiennes estimé récursivement est un cas particulier du mélange d'AR.

– Chaque temps est associé à un jeu de paramètres θ_t à estimer. La fonction de vraisemblance (3.89), conditionnée par θ_t, est de nouveau utilisée, mais augmentée d'une fonction de vraisemblance spécifique aux paramètres, la fonction d'évolution des paramètres. Celle-ci fait intervenir l'hypothèse de Markov :

$$p(\theta_t|\theta_{t-1}) \tag{3.90}$$

A chaque étape, on dispose de l'a posteriori de l'étape précédente $\tilde{p}(\theta_{t-1}|\mathbf{X}_{t-1})$. La mise à jour est alors décomposée en 2 étapes : l'évolution des paramètres, et la mise à jour temporelle. Le filtre de Kalman est une instance de ce modèle général, avec une gaussienne pour modéliser chaque vraisemblance [Kalman, 1960]. Ce dernier modèle est un cas particulier de filtrage bayésien variationnel [Smidl and Quinn, 2006, chap. 7.2].

Dans ce dernier cas, l'utilisation de distributions variationnelles factorisées permet d'estimer facilement les a posteriori $\tilde{p}(\theta_t|\mathbf{X}_t)$ et $\tilde{p}(\theta_{t-1}|\mathbf{X}_t)$, ouvrant ainsi la possibilité de boucles de rétro-action, pour réaliser à la fois du filtrage et du lissage.

Processus gaussiens

Une méthode alternative d'estimation bayésienne du mélange de gaussiennes est proposée dans [Rasmussen, 2000]. Celle-ci est inspirée de la régression par les processus gaussiens [Williams and Rasmussen, 1996]. Dans cette méthode, un modèle analogue à celui résumé par le schéma 3.6 est utilisé.

Dans un premier temps, un nombre fini de composantes gaussiennes est considéré, et les distributions a posteriori de chaque paramètre et variable latente, conditionnées par toutes les autres inconnues, sont calculées, sans passer par l'approximation variationnelle.

Ensuite, en faisant tendre K vers $+\infty$, les auteurs montrent qu'il suffit de connaitre les quelques composantes affectées par des éléments de données pour maintenir des estimateurs du modèle. L'algorithme d'estimation est alors le suivant :

– le modèle est initialisé à une seule composante,
– celui-ci évolue suivant la méthode d'échantillonnage de Gibbs, i.e. chaque paramètre est mis à jour à tour de rôle par échantillonnage selon sa distribution conditionnée par tous les autres.

Ces itérations définissent une chaine de Markov, dont on peut prouver que la distribution stationnaire est le modèle a posteriori recherché [Bishop, 2006, chap. 11].

Alternative pour l'estimation du rang de l'ACP probabiliste

Dans notre version bayésienne variationnelle de l'ACP probabiliste, et, par extension, des mélanges de ces modèles, nous avons proposé une méthode pour déterminer automatiquement le rang des matrices de facteurs de chaque composante (voir section 3.5.4).

Pour une ACP à une seule composante, la méthode traditionnelle est de choisir selon un seuil prédéfini sur les valeurs propres de la matrice de covariance empirique (i.e. le rang j à partir duquel $\lambda_{j+i} < \textit{seuil}$).

Une technique alternative d'inspiration bayésienne a été proposée [Minka, 2000], elle aussi se basant sur les seules valeurs propres de covariance. Une modélisation bayésienne du problème est définie en s'inspirant du formalisme présenté dans la section 3.5.4. Après avoir décliné une expression de la vraisemblance marginale conditionnée par la complexité du modèle $p(\mathbf{Y}|q)$, une approximation de celle-ci est calculée en utilisant la méthode de Laplace (à l'instar du critère BIC). Il en résulte un critère peu coûteux à calculer, et adapté au contexte de l'ACP.

Allocation Latente de Dirichlet

L'allocation latente de Dirichlet (ou LDA, pour *Latent Dirichlet Allocation*) est un modèle génératif de données discrètes, qui peut être assimilé à un mélange de multinomiales [Blei et al., 2003]. La terminologie des corpus de documents est utilisée pour la présentation de ce modèle : il peut donc être appliqué à des collections de documents textuels, mais n'est pas restreint a priori à cette application. Dans [Blei et al., 2003], une utilisation dans le contexte du filtrage collaboratif est notamment présentée.

Ce modèle, de nature probabiliste, peut donc être résumé par un DAG (figure 3.21).

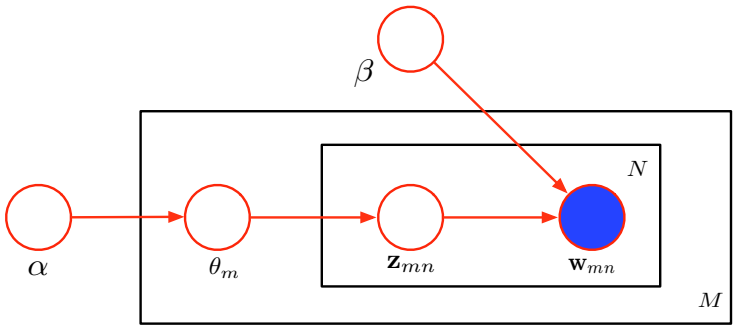


FIG. 3.21 – DAG pour le modèle LDA

Ce modèle adopte la représentation *bag-of-words* (i.e. l'ordre des mots dans le document n'est pas pris en compte). Chaque mot peut prendre sa valeur dans un vocabulaire

V ; le mot v_i est alors représenté par un vecteur 1-parmi-$|V|$ (voir définition (3.16)). Avec ce modèle, on suppose que les N mots de M documents sont générés par le modèle de la manière suivante :

- Chaque document $\mathbf{w}$ est associé à une distribution multinomiale parmi k thèmes. Cette distribution est obtenue par une loi de Dirichlet $\mathrm{Dir}(\theta|\alpha)$.
- Chaque mot w du document est conditionné par un thème z obtenu selon θ, et par le paramètre β. Ce dernier paramètre est une matrice spécifiant les probabilités de chaque mot du vocabulaire selon le thème associé.

Ainsi ce modèle permet une flexibilité importante, dans une approche probabiliste bien posée. En particulier, un document peut facilement être associé à plusieurs thèmes, ce qui n'est pas vrai pour les approches apparentées existantes.

Plusieurs variantes algorithmiques pour l'estimation d'un tel modèle à partir d'un corpus de documents sont considérées : les auteurs proposent d'abord un algorithme EM classique, puis, en définissant des distributions a priori sur une partie des paramètres, un algorithme partiellement variationnel est proposé. Celui-ci combine l'utilisation du théorème 3.5.1 pour déterminer les expressions de mise à jour des distributions variationnelles, et l'optimisation convexe des hyper-paramètres pour lesquels aucune distribution variationnelle n'a été définie.

3.6 Conclusion

Nous avons décrit en détail les fondements de l'approche génératrice à l'apprentissage automatique. Les modèles de mélange ont été introduits, ainsi que des méthodes pour leur estimation. Le principe variationnel bayésien a été détaillé, autorisant par la même occasion une synthèse de la littérature de ce sujet. À titre d'exemple, nous avons décrit l'implémentation de ce principe dans le cas du mélange de gaussiennes. Celle-ci est essentiellement tirée de [Bishop, 2006, chap. 10]. Nous proposons quelques résultats expérimentaux, soulignant les caractéristiques de l'approche variationnelle relativement à l'algorithme EM. Nous illustrons également avec le cas du mélange de MPPCA. Cet exemple est inspiré de [Beal, 2003], avec quelques adaptations théoriques, et des détails algorithmiques de notre fait. Nous avons souligné les propriétés de parcimonie présentées par les solutions produites par ces algorithmes.

Chapitre 4

Agrégation de modèles de mélange

4.1 Introduction

Dans ce chapitre, nous traitons de l'agrégation de modèles de mélange tels que ceux présentés dans le chapitre 3. Nous reprendrons donc beaucoup d'éléments évoqués dans le chapitre précédent, dont l'apprentissage variationnel, les mélanges de gaussiennes et les MPPCA. L'agrégation relève de l'apprentissage sur des données distribuées, dont nous avons donné un point de vue général dans la section 2.6.

Après avoir donné un état de l'art spécifique à la tâche d'agrégation, nous présentons quelques unes de nos propositions, ayant fait l'objet de contributions personnelles : une technique d'agrégation adaptée successivement au mélange de gaussiennes [Bruneau et al., 2008a,c, 2010a] et au MPPCA [Bruneau et al., 2010c,b].

Une approximation asymptotique est utilisée pour adapter les algorithmes VBGMM (voir section 3.5.2) et VBMPPCA (voir section 3.5.4), à l'origine dédiés au traitement d'éléments de données dans $\mathbb{R}^d$, à des entrées constituées de composantes gaussiennes.

Les principes de parcimonie de l'approche variationnelle sont exploités pour obtenir une solution de complexité adéquate. Enfin, nous présentons une adaptation semi-supervisée, permettant de prendre une source de données (i.e. site) pour chaque composante en entrée [Bruneau et al., 2009a].

4.2 Agrégation de modèles génératifs

On pourrait très simplement considérer la somme pondérée d'un ensemble de mélanges de gaussiennes comme une solution à l'agrégation et à l'estimation distribuée ; mais ceci conduit généralement à un modèle de complexité inutilement élevée. L'objectif est plutôt de réaliser l'estimation d'un mélange parcimonieux à partir des modèles fournis par les sources distribuées.

Le modèle «réduit» devra toutefois restituer la densité du processus sous-jacent de la meilleure manière possible. Cette propriété de parcimonie est spécialement importante si de telles agrégations doivent se succéder dans le temps, comme cela peut se produire dans un système d'apprentissage à grande échelle.

Dans [Merugu and Ghosh, 2003] le problème est présenté comme une minimisation de divergence KL entre la somme pondérée des modèles reçus depuis les sites, et le modèle agrégé à produire. Des éléments sont échantillonnés depuis la somme pondérée, et servent à estimer un nouveau modèle avec une technique usuelle, comme l'algorithme EM adjoint d'un critère de complexité.

Une adaptation simple pour l'apprentissage semi-supervisé est proposée, en construisant séparément, sur les sites répartis, un modèle avec des données étiquetées et un autre avec les données sans étiquette. Le processus de fusion est alors réalisé en pondérant avec l'information a priori. Cette dernière approche peut être vue comme une adaptation distribuée de la proposition effectuée dans [Nigam et al., 2006].

Cependant, une méthode reposant sur l'échantillonage peut être très couteuse, en particulier si les données sont supportées par un espace à dimension élevée. Par la suite, nous proposerons une méthode n'utilisant que les paramètres du modèle redondant, sans procédé d'échantillonnage. Le coût de calcul et de transmission est ainsi limité à son minimum.

En pratique, notre méthode revient à la recherche d'une combinaison optimale parmi les composantes gaussiennes fournies en entrée. Nous posons le problème sous une forme bayésienne, et proposons un algorithme EM variationnel pour la résolution, avec les propriétés de parcimonie évoquées dans la section 3.5.1.

Pour des problèmes de taille réduite, la recherche gourmande parmi toutes les combinaisons de composantes possible peut être réalisée via la méthode Hongroise, réalisant ainsi un optimum global [Kuhn, 1955, Korte and Vygen, 2002].

Des optima locaux à moindre coût sont proposés dans [Goldberger and Roweis, 2004], où les auteurs utilisent un critère basé sur la perte de divergence KL. Leur technique peut être assimilée à un algorithme K-means appliqué sur un ensemble de composantes. Alternativement, des procédures inspirées de la classification ascendante hiérarchique agissant sur un ensemble de composantes ont été proposées [Runnalls, 2006, Garcia et al., 2010, Nielsen and Boltz, 2010, Garcia and Nielsen, 2010].

Dans [Vasconcelos, 2001], les auteurs traitent un plus grand espace de recherche, en considérant également les combinaisons linéaires de composantes, plutôt que des affectations binaires. Ces combinaisons sont alors utilisées pour la construction de hiérarchies de mélanges de gaussiennes. Toutefois, ces derniers travaux ne traitent pas d'un critère et d'une procédure pour la détermination de la complexité souhaitable pour le modèle agrégé.

Pour situer la portée de nos travaux, nous pouvons contraster ceux-ci avec des

avancées en termes de combinaisons de classifieurs [Kittler et al., 1998], ou méthodes d'ensemble [Dietterich, 2000, Freund, 2001]. Cette approche générique peut-être envisagée de plusieurs manières :

- d'un point de vue bayésien, en estimant un ensemble de fonctions discriminantes probabilistes (e.g. régression logistique [Bishop, 2006, chap. 10.6]). À la présentation d'un individu, sa classe est déterminée selon la majorité des «votes» des différents modèles,

- d'un point de vue «mélange d'experts» [Jordan and Jacobs, 1994, Bishop and Svensén, 2003], en associant une fonction discriminante variable selon l'espace des données observées. Cette approche est basée sur les mélanges de gaussiennes conditionnels ([Bishop, 2006, chap. 14], voir section 3.3.3). Une structure hiérarchique (*Hierarchical Mixture of Experts*) est estimée, soit au moyen d'une combinaison des algorithmes EM et IRLS (*Iterated Reweighted Least Squares*), soit avec une approche variationnelle. Celle-ci permet «d'aiguiller», selon le domaine des données, vers le modèle de classification le mieux adapté,

- en manipulant un ensemble d'apprentissage de manière à entrainer un éventail de fonctions discriminantes à combiner par la suite, selon un système de vote dépendant des manipulations réalisées sur l'ensemble d'apprentissage (AdaBoost, Bagging [Dietterich, 2000]).

La méthode de combinaison de partitions, présentée dans [Strehl and Ghosh, 2003], et déjà brièvement évoquée dans la section 2.6, peut être affiliée à la philosophie des méthodes d'ensemble.

Dans ce chapitre, nous allons plutôt montrer comment des modèles de mélange (gaussiennes ou MPPCA) peuvent être simplifiés en adaptant les algorithmes VBGMM et VBMPPCA à des entrées constituées de composantes en lieu et place d'individus définis sur un espace vectoriel. La figure 4.1 propose une illustration simple de l'objectif recherché.

Une architecture de réseaux social distribuée est présentée dans [Kermarrec, 2009], et les techniques présentées dans ce chapitre pourraient y être intégrées. Dans le contexte des réseaux de capteurs, l'utilisation de statistiques synthétiques partagées, au lieu de paramètres de modèles dans le présent chapitre, est défendue dans [Gu, 2008]. Dans [Safarinejadian et al., 2010], une approche similaire est utilisée, conjointement avec une approche variationnelle pour l'estimation de mélanges de gaussiennes.

4.3 Aggrégation variationnelle de GMM

Dans cette section nous présentons une technique pour réaliser l'agrégation de mélanges de gaussiennes. Celle-ci a fait l'objet de quelques contributions personnelles [Bruneau et al., 2008a, 2010a], ainsi que d'une adaptation pour la structuration d'une

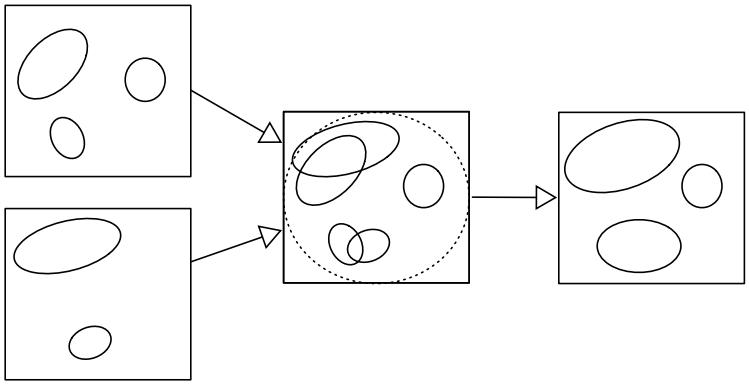


FIG. 4.1 – Exemple simple d'agrégation. Un a priori est indiqué en pointillés.

collection d'images selon des critères spatiaux et temporels [Bruneau et al., 2008c].

4.3.1 Représentation limite d'un échantillon virtuel

Dans la section 4.2, nous avons évoqué des problématiques liées aux modèles probabilistes de données distribuées, pour lesquelles l'estimation d'un modèle parcimonieux à partir de paramètres captés sur un réseau est une tâche cruciale.

Considérons un mélange connu de L composantes gaussiennes, avec comme paramètres $\theta' = \{\Omega', \mu', \Lambda'\}$. Notons que ce mélange peut être redondant (e.g. comme peut l'être un regroupement de modèles obtenus depuis plusieurs sources).

Supposons que des échantillons $\mathbf{X}$ et $\mathbf{Z}'$ ont été échantillonnés i.i.d selon cette distribution. Il est alors possible de regrouper $\mathbf{X}$ et $\mathbf{Z}'$ selon la composante dont ils sont originaires. Cela nous permet de poser le formalisme suivant : $\mathbf{X} = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_L\}$ avec $\text{card}(\mathbf{X}) = N, \hat{\mathbf{x}}_l = \{\mathbf{x}_n | z'_{nl} = 1\} = \{\mathbf{x}_{ln}\}$ et $\text{card}(\hat{\mathbf{x}}_l) = \omega'_l N$.

Considérons un nouveau mélange de gaussiennes inconnu à estimer selon le formalisme présenté dans la section 3.5.2, et exprimons la vraisemblance de l'échantillon $\{\mathbf{X}, \mathbf{Z}'\}$ que nous venons de définir selon l'expression (3.43). Notons que ce nouveau modèle associe à $\mathbf{X}$ une nouvelle variable latente $\mathbf{Z} \neq \mathbf{Z}'$.

Afin d'assurer des expressions simples, nous supposerons $\forall \mathbf{x}_n \in \hat{\mathbf{x}}_l, \mathbf{z}_n = \text{const} = \mathbf{z}_l$, ou autrement dit que tous les échantillons hypothétiques provenant d'une composante fournie particulière seront nécessairement affectés à la même composante du modèle à estimer. Cela peut sembler une hypothèse forte, mais comme la simplification d'un mélange revient souvent à regrouper des composantes, celle-ci est souvent valide. Nous pouvons alors ré-écrire (3.43) avec notre nouvelle définition pour l'échantillon $\mathbf{X}$:

$$p(\mathbf{X}|\mathbf{Z},\mu,\Lambda) = \prod_{k=1}^{K}\prod_{l=1}^{L} p(\hat{\mathbf{x}}_l|\mu_k,\Lambda_k)^{z_{lk}} \tag{4.1}$$

$$p(\mathbf{X}|\mathbf{Z},\mu,\Lambda) = \prod_{k=1}^{K}\prod_{l=1}^{L} \left[\prod_{n=1}^{\omega'_l N} \mathcal{N}(\mathbf{x}_{ln}|\mu_k,\Lambda_k^{-1})\right]^{z_{lk}} \tag{4.2}$$

$$\ln p(\mathbf{X}|\mathbf{Z},\mu,\Lambda) = \sum_{k=1}^{K}\sum_{l=1}^{L} z_{lk}\left[\sum_{n=1}^{\omega'_l N} \ln\mathcal{N}(\mathbf{x}_{ln}|\mu_k,\Lambda_k^{-1})\right] \tag{4.3}$$

Pour N suffisamment grand, nous pouvons faire l'approximation suivante :

$$\sum_{n=1}^{\omega'_l N} \ln\mathcal{N}(\mathbf{x}_{ln}|\mu_k,\Lambda_k^{-1}) \simeq \omega'_l N \mathbb{E}_{\mu'_l,\Lambda'_l}\left[\ln\mathcal{N}(\mathbf{x}|\mu_k,\Lambda_k^{-1})\right] \tag{4.4}$$

En clair, nous faisons l'approximation d'un *échantillon virtuel* à partir d'une espérance. Cette approximation a été introduite dans [Vasconcelos and Lippman, 1998, Vasconcelos, 2001].

Transformons l'espérance dans l'expression (4.4) :

$$\mathbb{E}_{\mu'_l,\Lambda'_l}\left[\ln\mathcal{N}(\mathbf{x}|\mu_k,\Lambda_k^{-1})\right] = \int \mathcal{N}(\mathbf{x}|\mu'_l,\Lambda_l'^{-1}) \ln\mathcal{N}(\mathbf{x}|\mu_k,\Lambda_k^{-1})\ \mathrm{d}\mathbf{x} \tag{4.5}$$

$$\mathbb{E}_{\mu'_l,\Lambda'_l}\left[\ln\mathcal{N}(\mathbf{x}|\mu_k,\Lambda_k^{-1})\right] = -\mathrm{KL}\left(\mathcal{N}(\mathbf{x}|\mu'_l,\Lambda_l'^{-1}) \parallel \mathcal{N}(\mathbf{x}|\mu_k,\Lambda_k^{-1})\right) - \mathrm{H}\big(\mathcal{N}(\mathbf{x}|\mu'_l,\Lambda_l'^{-1})\big) \tag{4.6}$$

avec $\mathrm{KL}(q_0 \parallel q_1)$ la divergence KL de q_1 par rapport à q_0 et $\mathrm{H}(q_0)$ l'entropie de q_0. Ces deux termes admettent une expression analytique [Blahut, 1987]. En ré-injectant (4.6) dans (4.4), puis (4.4) dans (4.3), nous obtenons l'expression suivante pour la log-vraisemblance de $\mathbf{X}$ conditionnée par $\mathbf{Z}$:

$$\ln p(\mathbf{X}|\mathbf{Z},\mu,\Lambda) = N\sum_{k=1}^{K}\sum_{l=1}^{L} z_{lk}\omega'_l\Big[-\mathrm{KL}\big(\mathcal{N}(\mathbf{x}|\mu'_l,\Lambda_l'^{-1})||\mathcal{N}(\mathbf{x}|\mu_k,\Lambda_k^{-1})\big) - H(\mathcal{N}(\mathbf{x}|\mu'_l,\Lambda_l'^{-1}))\Big] \tag{4.7}$$

$$\ln p(\mathbf{X}|\mathbf{Z},\mu,\Lambda) = N\sum_{k=1}^{K}\sum_{l=1}^{L} z_{lk}\omega'_l\Big[\frac{1}{2}\ln\det\Lambda_k - \frac{1}{2}\mathrm{Tr}(\Lambda_k\Lambda_l'^{-1}) - \frac{1}{2}(\mu'_l-\mu_k)^T\Lambda_k(\mu'_l-\mu_k) - \frac{d}{2}\ln(2\pi)\Big] \tag{4.8}$$

Remarquons qu'en définissant un échantillon hypothétique $\mathbf{X}$ originaire d'un mélange de gaussiennes connu θ', il nous a été possible de trouver une expression limite pour

$\ln p(\mathbf{X}|\mathbf{Z}, \mu, \Lambda)$ qui ne dépend pas de $\mathbf{X}$ ou de $\mathbf{Z}'$, mais seulement des paramètres du modèle connu fourni en entrée.

Notre changement de formalisme a également des conséquences sur (3.44) ; comme nous avons supposé précédemment que $z_{lk} = z_{nk}\ \forall \mathbf{x}_n \in \hat{\mathbf{x}}_l$, il est possible d'écrire :

$$p(\mathbf{Z}|\omega) = \prod_{n=1}^{N}\prod_{k=1}^{K} \omega_k^{z_{nk}} = \prod_{l=1}^{L}\prod_{k=1}^{K} \omega_k^{N\omega'_l z_{lk}} \tag{4.9}$$

Nous constatons que les expressions (4.8) et (4.9) dépendent désormais de L individus effectifs (1 par composante). Les équations de mise à jour de VBGMM dépendent de ces distributions modifiées, en particulier de moments pris selon celles-ci. Nous donnons ci-après les nouveaux moments et hyper-paramètres a posteriori en découlant, menant ainsi à un algorithme modifié, que nous nommerons VBGMM-A.

4.3.2 Nouveaux hyper-paramètres a posteriori

Conséquemment à l'utilisation de (4.8) et (4.9), l'expression (3.52) est adaptée à notre nouveau contexte, définissant de nouveaux estimateurs ρ_{lk} :

$$\begin{aligned}\ln(\rho_{lk}) = \frac{N\omega'_l}{2}\Bigg(&2\mathbb{E}[\ln \omega_k] + \mathbb{E}[\ln\det\Lambda_k] - d\ln(2\pi) \\ &- \mathbb{E}_{\mu_k,\Lambda_k}\left[\mathrm{Tr}(\Lambda_k\Lambda_l'^{-1}) + (\mu'_l - \mu_k)^T\Lambda_k(\mu'_l - \mu_k)\right]\Bigg)\end{aligned} \tag{4.10}$$

Ceux-ci sont utilisés de la même manière que dans VBGMM pour calculer les estimateurs r_{lk}. μ_k et Λ_k sont distribuées selon la même forme fonctionnelle que dans le cadre de l'algorithme VBGMM ; le moment (3.66) est donc facilement ré-évalué, donnant $\frac{d}{\beta}_k + \nu_k\left[\mathrm{Tr}(\mathbf{W}_k\Lambda_l'^{-1}) + (\mu'_l - \mathbf{m}_k)^T\mathbf{W}_k(\mu'_l - \mathbf{m}_k)\right]$.

De manière analogue à VBGMM, nous définissons un ensemble de statistiques synthétiques :

$$N_k = \sum_{l}^{L} N\omega'_l r_{lk} \tag{4.11}$$

$$\bar{\mathbf{x}}_k = \frac{1}{N_k}\sum_{l}^{L} N\omega'_l r_{lk}\mu'_l \tag{4.12}$$

$$\mathbf{S}_k = \frac{1}{N_k}\sum_{l}^{L} N\omega'_l r_{lk}(\mu'_l - \bar{\mathbf{x}}_k)(\mu'_l - \bar{\mathbf{x}}_k)^T \tag{4.13}$$

$$\mathbf{C}_k = \frac{1}{N_k}\sum_{l}^{L} N\omega'_l r_{lk}\Lambda_l'^{-1} \tag{4.14}$$

La formule (3.41) est appliquée pour $q(\omega)$ et $q(\mu, \Lambda)$ dans ce nouveau contexte ; rappelons que les distributions utilisées sont toujours de la même forme fonctionnelle que dans le cas de VBGMM. Les hyper-paramètres de l'a posteriori conjugué existent donc, et sont donnés par :

$$\alpha_k = \alpha_0 + N_k \tag{4.15}$$
$$\beta_k = \beta_0 + N_k \tag{4.16}$$
$$\mathbf{m}_k = \frac{1}{\beta_k}\left(\beta_0 \mathbf{m}_0 + N_k \bar{\mathbf{x}}_k\right) \tag{4.17}$$
$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k \mathbf{S}_k + N_k \mathbf{C}_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\bar{\mathbf{x}}_k - \mathbf{m}_0)(\bar{x}_k - \mathbf{m}_0)^T \tag{4.18}$$
$$\nu_k = \nu_0 + N_k \tag{4.19}$$

4.3.3 Mesure de convergence et paramétrage

L'algorithme VBGMM diminue monotoniquement la divergence KL entre la distribution variationnelle optimale et la vraie distribution a posteriori inconnue (voir section 3.5.1). Cette diminution est équivalente à l'augmentation monotonique d'une borne inférieure à la vraisemblance marginale. En calculant celle-ci, on dispose alors d'un moyen pour tester la convergence de l'algorithme. L'expression complète de celle-ci peut-être trouvée dans [Bishop, 2006, chap. 10].

Notre changement de formalisme revenant au replacement de $(\mathbf{X}, \mathbf{Z}')$ par une valeur limite, seuls les termes de borne inférieure impliquant $\mathbf{X}$ et $\mathbf{Z}$ sont changés. Nous donnons ici les seuls termes modifiés, devant être utilisés en lieu et place de leurs termes originaux respectifs :

$$\mathbb{E}[\ln p(\mathbf{X}|\mathbf{Z}, \mu, \Lambda)] = \frac{1}{2}\sum_k N_k \Big\{ \ln \tilde{\Lambda}_k - \frac{d}{\beta_k} - \nu_k \mathrm{Tr}\big((\mathbf{S}_k + \mathbf{C}_k)\mathbf{W}_k\big) - \nu_k (\bar{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\bar{\mathbf{x}}_k - \mathbf{m}_k) - d\ln(2\pi) \Big\} \tag{4.20}$$
$$\mathbb{E}[\ln p(\mathbf{Z}|\omega)] = \sum_k N_k \ln \tilde{\omega}_k \tag{4.21}$$

Nous proposons de paramétrer VBGMM-A exactement comme VBGMM. Dans le cas de ce dernier algorithme, dans la section 3.5.2 nous avons vu que la paramétrage utilisé permettait d'assurer la parcimonie de l'a posteriori estimé. Dans le cas de l'algorithme VBGMM-A, nous cherchons la parcimonie d'un échantillon virtuel ; cela revient donc à supprimer la redondance, ou réduire, le mélange de gaussiennes fourni en entrée.

Dans la section suivante, nous proposons quelques expériences illustrant l'utilisation possible de cette technique.

4.4 Expérimentation de VBGMM-A

Cette partie expérimentale est une version développée de notre article [Bruneau et al., 2008a].

4.4.1 Protocole

Le protocole expérimental présenté dans cette section est originellement inspiré des expériences réalisées dans [Vasconcelos, 2001]. Pour celles-ci, la base de données d'objets Columbia (COIL [Nene et al., 1996]) a été utilisée. Cette base contient 100 objets, avec 72 vues différentes pour chacun. Les vues sont séparées de 5°. Nous sélectionnons 9 vues séparées chacune de 40° pour chaque objet.

Nous utilisons le processus d'extraction de caractéristiques présenté dans [Vasconcelos, 2001] (en bref, qui revient à l'extraction de coefficients DCT par blocs). Pour les présentes expériences, nous ne retenons que les 12 premières variables, afin d'éviter les problèmes impliqués par l'usage d'espaces à très hautes dimensionalités.

Pour chaque image, les données extraites servent à l'estimation d'un mélange de gaussiennes en utilisant l'algorithme VBGMM 2, au lieu de l'algorithme EM classique dans [Vasconcelos, 2001]. Le nombre de composantes adéquat pour le représentant de chaque image est donc choisi automatiquement.

Pour chaque objet, la première vue forme la requête, et les 8 restantes forment un groupe de référence pour l'objet associé. Chaque groupe est résumé en regroupant toutes les composantes de ses représentants, et en effectuant la réduction de ce regroupement avec l'algorithme VBGMM-A. De nouveau, la complexité de ces réductions est déterminée automatiquement.

En utilisant ce modèle résumé, seulement une mesure de similarité sera nécessaire pour estimer le groupe associé à une requête. On mesure la similarité entre le modèle requête q et un des modèles de référence p avec $\mathrm{KL}(q||p)$ (voir (3.32) pour une définition). Celle-ci est estimée simplement au moyen de simulations Monte-Carlo.

4.4.2 Résultats

L'apprentissage des résumés de groupes est très rapide (entre 5 et 10 itérations en général), et la complexité du calcul ne dépend que de la dimensionalité des données d, et du nombre de modèles agrégés. Ces valeurs sont typiquement modestes (dans le cas présent $d = 12$ et $N_{\text{modèles}} = 8$) au regard du nombre de pixels impliqués.

Nous obtenons 75% de succès pour la reconnaissance de l'objet exact tentant d'associer une des requêtes avec un modèle agrégé de groupe. Plus généralement, 95% des 3 modèles les plus proches d'une requête (i.e. avec la divergence KL la plus basse)

contiennent la bonne réponse.

Nous obtenons donc des résultats similaires à ceux présentés dans [Vasconcelos, 2001], sachant que chaque représentant d'image et de groupe était alors contraint à $K = 8$. Notre approche est plus parcimonieuse (voir figure 4.2), et le coût et les contraintes algorithmiques impliquées sont moindres (voir la section 3.4.4 pour des détails à ce sujet). Notons que tous les représentants ont été construits de manière non-supervisée, avec des initialisations aléatoires.

Les algorithmes EM variationnels nous permettent d'obtenir des solutions plus satisfaisantes, mais la nature locale de l'estimateur obtenu rend les solutions estimées instables (notamment à l'initialisation). Ainsi, les représentants servant à la construction de nos modèles de groupes peuvent être passablement bruités. Dans beaucoup de situations cette influence peut n'être que marginale, mais dans certains cas, il peut être souhaitable de réaliser plusieurs estimations de représentants avec VBGMM, et de conserver celle qui amène la meilleure valeur de borne inférieure.

taux de succès	75 %
succès parmi les 3 1^{ers} résultats	95 %
nombre de composantes moyen / représentant d'image	6.05
nombre de composantes moyen / modèle de groupe	2.06

FIG. 4.2 – Résultats expérimentaux

4.5 Aggrégation variationnelle de MPPCA

Cette section a fait l'objet de contributions personnelles [Bruneau et al., 2010b,c].

Supposons que l'on dispose désormais d'un ensemble de MPPCA, que l'on souhaite agréger. Ceux-ci sont obtenus soit au moyen de l'algorithme VBMPPCA 4, ou de mélanges de gaussiennes, obtenus par VBGMM, dont les matrices de covariance auront été préalablement diagonalisées.

La manière la plus naïve d'agréger ces MPPCA estimés séparément est d'en faire la somme pondérée. Cependant, si chaque source de données ayant donné lieu à un modèle reflète le même processus sous-jacent, le mélange résultant comprendra un nombre superflu de composantes ; on constatera de nombreuses redondances en comparaison au modèle qui aurait été estimé sur la réunion des sources.

Dans cette section, nous utilisons de nouveau le principe d'échantillonnage virtuel, cette fois adapté au MPPCA. Nous incorporons ensuite une représentation limite dans l'algorithme 4 en lieu et place d'un échantillon ordinaire. L'exécution de l'algorithme nous permet ainsi d'obtenir le modèle réduit le plus représentatif de l'échantillon qui

aurait été généré par l'entrée redondante, sans utiliser de données originales, ni aucun procédé d'échantillonnage.

4.5.1 Vraisemblance d'un échantillon virtuel

Un échantillon complet (i.e. les données et les étiquettes) provenant d'un MPPCA arbitraire et connu de L composantes peut être noté $(\mathbf{Y}, \mathbf{Z}') = \{\mathbf{y}_n, \mathbf{z}'_n\}$, avec la notation 1-parmi-L habituelle pour les $\mathbf{z}'_n$ (voir définition 3.16). De manière analogue au raisonnement effectué dans la section 4.3.1, les éléments de cet échantillon peuvent être regroupés selon les valeurs prises par $\mathbf{z}'_n$: nous notons $\hat{\mathbf{y}}_l = \{\mathbf{y}_n | z'_{nl} = 1\}$ et $\hat{\mathbf{z}}'_l = \{\mathbf{z}'_n | z'_{nl} = 1\}$.

Nous allons exprimer la vraisemblance de cet échantillon selon un nouveau MPPCA inconnu, et proposer un algorithme d'apprentissage pour celui-ci. Ce dernier sera désormais nommé le modèle *cible*, ou de sortie. Afin d'éviter toute ambiguité, nous rappelons les conventions d'indexation et notations compactes appropriées ; les composantes du mélange d'entrée et de sortie seront indexées respectivement par $l \in \{1 \dots L\}$ et $k \in \{1 \dots K\}$.

Nous avons vu que les étiquettes du modèle d'entrée sont résumées par $\mathbf{Z}'$: celles du modèle de sortie le seront par $\mathbf{Z}$. Le regroupement des composantes en entrée sera collectivement dénoté par θ', et le modèle de sortie par θ. Aussi, afin de conserver les expressions les plus claires possibles, et sans perte d'expressivité, nous supposerons que toutes les composantes du mélange d'entrée ont le même nombre de facteurs, q.

Le modèle de sortie n'est pas censé être plus complexe que l'entrée, aussi ce même q pourra être utilisé pour paramétrer le modèle de sortie. La vraisemblance complète du modèle de sortie selon notre échantillon est alors donnée par :

$$p(\mathbf{Y}, \mathbf{Z}|\theta) = p(\mathbf{Y}|\mathbf{Z}, \theta)p(\mathbf{Z}|\theta) \tag{4.22}$$

$$\text{avec } p(\mathbf{Y}|\mathbf{Z}, \theta) = \prod_k \prod_l (\mathcal{N}(\hat{\mathbf{y}}_l|\mu_k, \mathbf{C}_k))^{z_{lk}} \tag{4.23}$$

$$\text{and } p(\mathbf{Z}|\theta) = \prod_k \prod_l (\omega_k^{|\hat{\mathbf{z}}'_l|})^{z_{lk}} \tag{4.24}$$

où nous avons utilisé l'expression (3.75) dans le contexte d'un échantillon i.i.d. $\mathbf{Y}$. La variable latente $\mathbf{X}$ est pour le moment intentionnellement marginalisée (voir eqn. (3.74)). Ces formules sont valides sous l'hypothèse que les $\mathbf{z}_l$ sont identiques pour tout $\mathbf{y}_i$ dans $\hat{\mathbf{y}}_l$.

Dans la plupart des cas cette hypothèse est raisonnable, étant donné que l'agrégation de modèles de mélange revient le plus souvent à combiner des composantes.

Remarquons que :

$$|\hat{\mathbf{z}}'_l| = |\hat{\mathbf{y}}_l| \simeq N\omega_l \tag{4.25}$$

où N indique la taille de l'échantillon mentionné plus haut. En incorporant cette approximation, (4.24) ne dépend maintenant de l'échantillon qu'au travers de sa taille et des paramètres du modèle d'entrée.

(4.23) peut être ré-écrite comme suit :

$$p(\mathbf{Y}|\mathbf{Z},\theta) = \prod_k \prod_l (\mathcal{N}(\hat{\mathbf{y}}_l|\mu_k,\mathbf{C}_k))^{z_{lk}} = \prod_k \prod_l p_{lk}^{z_{lk}} \tag{4.26}$$

Maintenant considérons un terme p_{lk} en particulier. Nous en prenons d'abord le log (obtenant ainsi $\mathcal{L}_{lk}$), développons $\hat{\mathbf{y}}_l$, puis utilisons l'approximation (4.25) :

$$\mathcal{L}_{lk} \simeq \sum_j^{N\omega_l} \ln \mathcal{N}(\mathbf{y}_{lj}|\omega_k,\mu_k,\mathbf{C}_k) \tag{4.27}$$

$$\simeq N\omega_l \Big[-\mathrm{KL}\big(\mathcal{N}(\mu_l,\mathbf{C}_l)||\mathcal{N}(\mu_k,\mathbf{C}_k)\big) - \mathrm{H}\big(\mathcal{N}(\mu_l,\mathbf{C}_l)\big) \Big] \tag{4.28}$$

où nous avons utilisé la loi des grands nombres pour approcher la divergence KL et l'entropie par une somme finie. Aussi N devra être suffisamment grand afin d'assurer la validité des expressions (4.25) et (4.28).

Soulignons de nouveau un fait remarquable : à l'instar du développement présenté dans la section 4.3.1, notre expression de vraisemblance approchée ne dépend plus de l'échantillon, mais seulement de la loi l'ayant généré. N peut ainsi être choisi arbitrairement. En d'autres termes, nous avons défini l'expression limite de notre échantillon virtuel.

La divergence KL entre gaussiennes, et l'entropie d'une distribution gaussienne, ont des expressions analytiques. Cela nous permet de ré-écrire (4.28) :

$$\mathcal{L}_{lk} = N\omega_l \Big[-\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln\det(\Lambda_k\Lambda_k^T + \tau^{-1}\mathbf{I}_d) \\ - \frac{1}{2}\mathrm{Tr}\big((\Lambda_k\Lambda_k^T + \tau^{-1}\mathbf{I}_d)^{-1}[\Lambda_l\Lambda_l^T + \tau^{-1}\mathbf{I}_d + (\mu_l-\mu_k)(\mu_l-\mu_k)^T]\big) \Big] \tag{4.29}$$

Dans le reste du développement, nous négligeons l'influence de τ, car la valeur ML de ce terme reflète les plus petites valeurs propres des sous-espaces des composantes associées. Comme $\Lambda_l\Lambda_l^T = \sum_j \Lambda_l^j \Lambda_l^{jT}$, $\mathcal{L}_{lk}$ peut, de manière approchée, être vue comme la vraisemblance combinée des moyennes et des facteurs de nos composantes d'entrée (après renormalisation, et en utilisant comme moyennes respectives μ_k et $\mathbf{0}$) :

$$p_{lk} = \Big[\mathcal{N}(\mu_l|\mu_k,\mathbf{C}_k) \prod_j \mathcal{N}(\Lambda_l^j|\mathbf{0},\mathbf{C}_k) \Big]^{N\omega_l} \tag{4.30}$$

En utilisant (4.30) avec (4.26) et (4.24), on obtient une approximation pour la vraisemblance complète (i.e. conjointement de $\mathbf{Y}$ et $\mathbf{Z}$) de notre échantillon virtuel :

$$p(\mathbf{Y},\mathbf{Z}|\theta) = \prod_l \prod_k \Big[\omega_k^{N\omega_l}\Big(\mathcal{N}(\mu_l|\mu_k,\mathbf{C}_k)\prod_j \mathcal{N}(\Lambda_l^{:j}|0,\mathbf{C}_k)\Big)^{N\omega_l}\Big]^{z_{lk}} \tag{4.31}$$

Dans (4.31), tous les termes sont de la famille exponentielle. En prenant une puissance d'une telle distribution, le résultat est toujours dans la famille exponentielle ; aussi $N\omega_l$ peut être incorporé dans les paramètres. On obtient ainsi :

$$p(\mathbf{Y},\mathbf{Z}|\theta) = p(\mathbf{Z}|\theta)p(\theta'|\mathbf{Z},\theta) \tag{4.32}$$

Afin de garder (4.32) sous une forme assez concise, on a utilisé :

$$p(\mathbf{Z}|\theta) = \prod_l \prod_k \left(\omega_k^{N\omega_l}\right)^{z_{lk}} \tag{4.33}$$

$$p(\theta'|\mathbf{Z},\theta) = \prod_l \prod_k \left(\mathcal{N}(\mu_l|\mu_k,(N\omega_l)^{-1}\mathbf{C}_k)\prod_j^q \mathcal{N}(\Lambda_l^{:j}|0,(N\omega_l)^{-1}\mathbf{C}_k)\right)^{z_{lk}} \tag{4.34}$$

Notons que par homogénéité avec l'algorithme VBMPPCA, nous continuons à évoquer l'échantillon virtuel $\mathbf{Y}$, même si celui-ci n'apparait plus dans les expressions.

En exploitant la représentation *1-parmi-K* de $\mathbf{Z}$, remarquons que (4.32) pourrait être transformée en mélange de produits de gaussiennes.

Les termes gaussiens de notre expression de vraisemblance dépendent toujours de $\mathbf{C}_k$, ils peuvent donc être étendus avec des variables latentes $\mathbf{x}$ en utilisant les propriétés des modèles linéaires gaussiens (voir formule (3.74)).

Dans l'approche classique au MPPCA [Beal, 2003, Tipping and Bishop, 1999b], on a une variable $\mathbf{x}$ par élément de l'échantillon. Maintenant, chaque composante en entrée est associée avec $1+q$ termes approchés, donc la taille de $\mathbf{X}$ augmente de la même manière.

Pour éviter les ambiguïtés parmi cet ensemble de variables latentes, nous définissons $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$, $\mathbf{X}_1 = \{\mathbf{x}_{1l}\}$ et $\mathbf{X}_2 = \{\mathbf{x}_{2lj} | j \in 1 \dots q\}$. L'expression de vraisemblance complète (i.e. des données complètes $\{\mathbf{Y}, \mathbf{X}_1, \mathbf{X}_2, \mathbf{Z}\}$) devient alors :

$$p(\mathbf{Y},\mathbf{X}_1,\mathbf{X}_2,\mathbf{Z}|\theta) = p(\mathbf{Z}|\theta)p(\theta'|\mathbf{Z},\mathbf{X}_1,\mathbf{X}_2,\theta)p(\mathbf{X}_1,\mathbf{X}_2|\mathbf{Z},\theta) \tag{4.35}$$

$$\text{avec } p(\theta'|\mathbf{Z},\mathbf{X}_1,\mathbf{X}_2,\theta) = \prod_l \prod_k \Big(\mathcal{N}\big(\mu_l|\Lambda_k\mathbf{x}_{1l}+\mu_k,(N\omega_l\tau)^{-1}\mathbf{I}_d\big)\cdot \prod_j^q \mathcal{N}\big(\Lambda_l^{:j}|\Lambda_k\mathbf{x}_{2lj},(N\omega_l\tau)^{-1}\mathbf{I}_d\big)\Big)^{z_{lk}}$$

$$\text{et } p(\mathbf{X}_1,\mathbf{X}_2|\mathbf{Z},\theta) = \prod_l \prod_k \left(\mathcal{N}\big(\mathbf{x}_{1l}|(N\omega_l)^{-1}\mathbf{I}_q\big)\prod_j^q \mathcal{N}\big(\mathbf{x}_{2lj}|(N\omega_l)^{-1}\mathbf{I}_q\big)\right)^{z_{lk}}$$

4.5.2 Borne inférieure associée

L'algorithme VBMPPCA réalise l'optimisation d'une borne inférieure à la vraisemblance marginale de l'échantillon (voir expression (3.85)), obtenant alors un estimateur du modèle MPPCA a posteriori.

Dans ce cadre, nous avons aussi défini une expression factorisée (3.84) pour la distribution variationnelle optimisée. Dans le cas présent, celle-ci est augmentée de termes associés aux variables latentes supplémentaires introduites dans notre expression de vraisemblance (4.35).

En conséquence, la borne inférieure (3.85) sera légèrement modifiée pour notre nouvel algorithme. La structure du MPPCA de sortie est inchangée, il en sera donc de même pour les 2 premières lignes de (3.85). Le terme sommé sur N relève de la vraisemblance des données, aussi l'expression (4.35) doit être utilisée en lieu et place du terme de vraisemblance standard. Ce terme (nommé $\mathcal{F}_{lk}$) est le suivant :

$$\begin{aligned}\mathcal{F}_{lk} = r_{lk}\Bigg(& -\ln r_{lk} + N\omega_l \mathbb{E}[\ln \omega_k] + \mathbb{E}\big[\ln p(\mathbf{x}_{1l}|z_{lk}=1)\big] + H\big[q(\mathbf{x}_{1l}|z_{lk}=1)\big] \\ & + \mathbb{E}\big[\ln p(\mu_l|z_{lk}=1,\mathbf{x}_{1l},\Lambda_k,\mu_k)\big] + \sum_j^q \Bigg(\mathbb{E}[\ln p(\mathbf{x}_{2lj}|z_{lk}=1)] \\ & + H[q(\mathbf{x}_{2lj}|z_{lk}=1)] + \mathbb{E}[\ln p(\Lambda_l^{\cdot j}|z_{lk}=1,\mathbf{x}_{2lj},\Lambda_k)] \Bigg) \Bigg) \end{aligned} \tag{4.36}$$

Où les expressions conditionnées par $z_{lk}=1$ sont obtenues en notant que $p(\mathbf{Z}|\theta)$ et $q(\mathbf{Z})$ suivent des lois multinomiales, et $r_{lk} = \mathbb{E}[z_{lk}]$.

Les mises à jour des paramètres variationnels peuvent être obtenues en appliquant la formule (3.41) conjointement à la vraisemblance modifiée (4.35), donnant :

– **Etape E** :

$$
\begin{aligned}
&\Sigma_{\mathbf{x}_{kl}} = \Big(N\omega_l[\mathbf{I}_q + \tau\mathbb{E}[\Lambda_k^T\Lambda_k]]\Big)^{-1} = (N\omega_l)^{-1}\Sigma_{\mathbf{x}_k}\\
&\mathbb{E}[\mathbf{x}_{1lk}] = \tau N\omega_l\Sigma_{\mathbf{x}_{kl}}\mathbb{E}[\Lambda_k^T]\Big(\mu_l - \mathbb{E}[\mu_k]\Big) = \tau\Sigma_{\mathbf{x}_k}\mathbb{E}[\Lambda_k^T]\Big(\mu_l - \mathbb{E}[\mu_k]\Big)\\
&\mathbb{E}[\mathbf{x}_{2lkj}] = \tau N\omega_l\Sigma_{\mathbf{x}_{kl}}\mathbb{E}[\Lambda_k^T]\Lambda_l^{\cdot j} = \tau\Sigma_{\mathbf{x}_k}\mathbb{E}[\Lambda_k^T]\Lambda_l^{\cdot j}\\
&r_{lk} \propto N\omega_l(\psi(\alpha_k) - \psi(\sum_j \alpha_j)) - \frac{N\omega_l}{2}\mathrm{Tr}(\mathbb{E}[\mathbf{x}_{1lk}\mathbf{x}_{1lk}^T] + \sum_j \mathbb{E}[\mathbf{x}_{2lkj}\mathbf{x}_{2lkj}])\\
&- \frac{\tau N\omega_l}{2}\Bigg[\|\mu_l\|^2 + \sum_j\|\Lambda_l^{\cdot j}\|^2 + \mathbb{E}[\|\mu_k\|^2] - 2\mu_l^T\mathbb{E}[\mu_k]\\
&- 2(\mu_l - \mathbb{E}[\mu_k])^T\mathbb{E}[\Lambda_k]\mathbb{E}[\mathbf{x}_{1lk}] - 2\sum_j(\Lambda_l^{\cdot jT}\mathbb{E}[\Lambda_k]\mathbb{E}[\mathbf{x}_{2lkj}])\\
&+ \mathrm{Tr}\Big(\mathbb{E}[\Lambda_k^T\Lambda_k]\big(\mathbb{E}[\mathbf{x}_{1lk}\mathbf{x}_{1lk}^T] + \sum_j\mathbb{E}[\mathbf{x}_{2lkj}\mathbf{x}_{2lkj}^T]\big)\Big)\Bigg]
\end{aligned}
\tag{4.37}
$$

– **Statistiques synthétiques** :

$$
\begin{aligned}
&N_k = \sum_l N\omega_l r_{lk} && \mathbf{y}_k = \sum_l N\omega_l r_{lk}\mu_l\\
&\mathbf{s}_k = \sum_l N\omega_l r_{lk}\mathbb{E}[\mathbf{x}_{1lk}] && \mathbf{sy}_k = \sum_l N\omega_l r_{lk}\Big(\mu_l\mathbb{E}[\mathbf{x}_{1lk}]^T\\
&&& \qquad + \sum_j \Lambda_l^{\cdot j}\mathbb{E}[\mathbf{x}_{2lkj}]^T\Big)\\
&\mathbf{S}_k = \sum_l N\omega_l r_{lk}\Big(\mathbb{E}[\mathbf{x}_{1lk}\mathbf{x}_{1lk}^T] && \mathbf{y}_k^2 = \sum_l N\omega_l r_{lk}\Big(||\mu_l||^2 + \sum_j||\Lambda_l^{\cdot j}||^2\Big)\\
&\qquad + \sum_j\mathbb{E}[\mathbf{x}_{2lkj}\mathbf{x}_{2lkj}^T]\Big)
\end{aligned}
\tag{4.38}
$$

– **Etape M** : le modèle estimé restant identique dans ce nouvel algorithme, en utilisant le nouvel ensemble de statistiques synthétiques les expressions de mise à jour (3.88) restent valides.

Ces nouvelles expressions nous permettent de définir l'algorithme VBMPPCA-A en tant qu'extension de l'algorithme 4, avec les même propriétés théoriques que celui-ci : monotonicité de la borne inférieure et obtention du MPPCA a posteriori.

4.5.3 Paramétrage

Dans la section 3.5.4, nous avons précisé nos choix de distributions a priori. Ces mêmes choix sont toujours valables, mais nous allons également exploiter la structure

des données désormais traitées.

Ainsi, au paragraphe 3.5.4, nous avons souligné que la procédure d'estimation des PPCA permettait de retrouver la matrice des vecteurs propres caractérisant chaque composante, ceux-ci étant ordonnés selon leur valeur propre (ou taille) décroissante. Remarquons maintenant que les variables latentes additionnelles peuvent être chacune associées à des colonnes des matrices Λ en entrée.

Sous l'hypothèse de facteurs Λ fournis dans un ordre correct, intuitivement nous pourrions associer la première colonne des facteurs Λ en entrée à la première colonne des Λ en sortie, etc... Comme les variables $\mathbf{x}$ indiquent une combinaison linéaire des colonnes des facteurs à estimer, nous choisissons donc d'initialiser les estimateurs pour $\mathbf{X}_2$ avec des vecteurs canoniques, afin de refléter cet a priori.

Nos expériences ont confirmé que la mise en oeuvre de ce principe était très efficace. La mise à jour des estimateurs de $\mathbf{X}_2$ alourdissent l'étape E de l'algorithme, mais cette perte est largement compensée si on remarque que désormais la complexité de notre algorithme dépend d'un nombre de composantes en entrée, et non de la taille d'un échantillon.

Au début de la section 4.5.1, nous avons affirmé que toutes les composantes, d'entrée ou de sortie, peuvent être associées au même paramètre q sans que cela pose de problème. Considérons maintenant le cas général, ou toutes les matrices de facteurs en entrée sont associées à un q_i spécifique. Posons $q_{\max} = \max q_i$. Nous voyons qu'en complétant chaque matrice de facteurs avec $q_{\max} - q_i$ colonnes nulles, et en forçant les variables $\mathbf{x}_{2..}$ associées à $\mathbf{0}$ au lieu d'un vecteur canonique revient à utiliser le même q pour toutes les matrices de facteurs.

4.6 Comparaison expérimentale de VBMPPCA-A et VBGMM-A

4.6.1 Echantillons de données

Nous présentons des résultats expérimentaux utilisant les jeux de données suivants :

- **Gaussian** : cet échantillon synthétique est obtenu en effectuant des tirages aléatoires depuis 3 gaussiennes 3D bien séparées (voir figure 4.3-2). 6 dimensions additionnelles ont été construites par combinaison linéaire aléatoire des variables du signal original, additionnées de bruit blanc. 2000 points par gaussienne sont générés.

- **semisphere** : cet échantillon synthétique a été obtenu par tirages uniformes d'angles selon une demi-sphère (voir figure 4.3-2). 6000 points sont générés.

– **circle** : 6000 points sont générés le long d'un cercle 2D, additionnés de bruit blanc (voir figure 4.3-3). Ce signal 2D a été transformé en 6D grâce à une matrice orthogonale aléatoire avec l'ajout, de nouveau, de bruit blanc.

– **Reconnaissance de chiffres manuscrits** (nommé par la suite **pen data**) : 10992 éléments définis sur 16 variables, construits à partir de mesures de position d'un stylet sur la tablette ayant servi à écrire des chiffres [Alimoglu, 1996]. Chaque élément est étiqueté avec le chiffre qui a été tracé pour sa construction (0-9, définissant ainsi 10 classes).

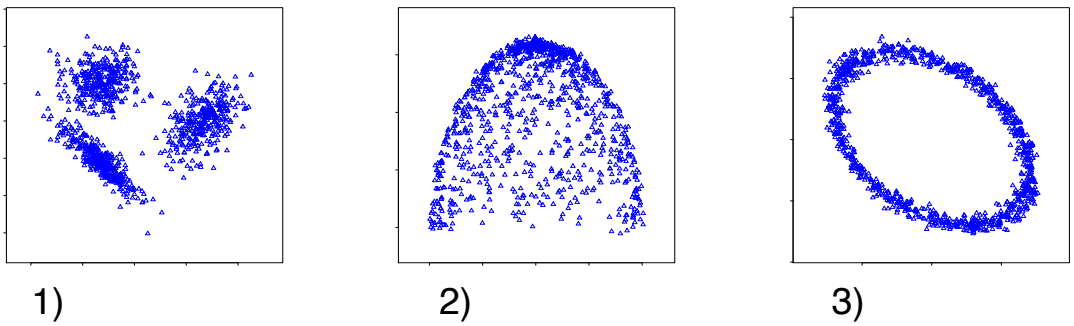


FIG. 4.3 – Echantillons synthétiques (projections 2D). 1) Gaussian 2) semisphere 3) circle

Notons que les échantillons utilisés ont des dimensionalités relativement modestes au regard d'une utilisation raisonnable d'ACP probabilistes. Toutefois, à fins de comparaison avec l'estimation via VBGMM, ce paramètre doit rester raisonnablement bas ; d'autre part, nos échantillons synthétiques présentent une vérité terrain clairement identifiée en ce qui concerne les sous-espaces latents. Toutes ces raisons expliquent nos choix d'échantillons.

4.6.2 Estimation de densités de probabilité

Nous évaluons ici la faculté de VBMPPCA-A à construire des modèles de densité depuis des sources distribuées, celles-ci ne transmettant que les paramètres de leur modèle local. À cette fin, les 3 échantillons synthétiques présentés précédemment sont utilisés dans le cadre du protocole suivant :

– des modèles d'entrée sont construits en utilisant l'algorithme VBMPPCA sur des sous-échantillons aléatoires de nos jeux de données. Tous les échantillons synthétiques contiennent 6000 éléments, et à chaque expérience 200 en sont extraits. 500 modèles sont ainsi construits, formant autant de sources distribuées parmi lesquelles nous choisirons nos entrées. A titre de comparaison, des mélanges de gaussiennes sont estimés sur les mêmes sous-échantillons en utilisant VBGMM.

– Nous utilisons l'algorithme VBMPPCA-A pour agréger un nombre variable n de sources de données, choisies aléatoirement parmi l'ensemble construit précédemment. Pour chaque valeur de n expérimentée, la procédure a été répétée 20 fois, et les résultats sont obtenus par une moyenne sur les mesures alors effectuées.

  L'agrégation des mêmes sources avec VBGMM-A nous donne une référence comparative. Pour que la comparaison soit équitable, les modèles à agréger sont choisis de manière identique parmi l'ensemble de MPPCA estimés sur nos sources pour les deux méthodes. Bien entendu, ceux-ci auront été convertis en mélanges de gaussiennes grâce à la formule (3.74) pour réaliser la fusion avec VBGMM-A.

Les mesures suivantes sont effectuées :

– Les complexités de nos modèles de sources (i.e. nombre de composantes et de facteurs par composante). Cela évalue la capacité de VBMPPCA à découvrir automatiquement ces caractéristiques. À titre de comparaison, le nombre de composantes découvert en utilisant VBGMM sur les mêmes sources est indiqué.

– La divergence JS des mélanges agrégés relativement à la somme pondérée des sources utilisées dans chaque expérience, en fonction de n. Cela indique la qualité de l'agrégation.

– Le nombre de composantes en sortie, en fonction de n.

	K' (MPPCA)	K' (mélange de gaussiennes)	q'
Gaussian	3.3	2.98	2.71
semi-sphere	13.8	11.16	1.64
circle	7.43	4.27	1.25

TAB. 4.1 – Complexités obtenues (nombre de composantes K', dimensionalité moyenne des sous-espaces de chaque composante q') pour les modèles MPPCA d'entrée utilisés dans nos expériences. Les sous-échantillons utilisés pour créer ces modèles servent également à estimer des mélanges de gaussiennes avec VBGMM à des fins de comparaison.

La table 4.1 résume les comportements respectifs de VBGMM et VBMPPCA. En moyenne, nous voyons que VBMPPCA tend à produire des modèles plus complexes. Mais le caractère problématique de cette observation n'est pas manifeste ; le nombre de composantes pour l'échantillon *Gaussian* semble sur-estimé avec VBMPPCA, alors que 7 composantes semblent mieux à même de modéliser *circle*.

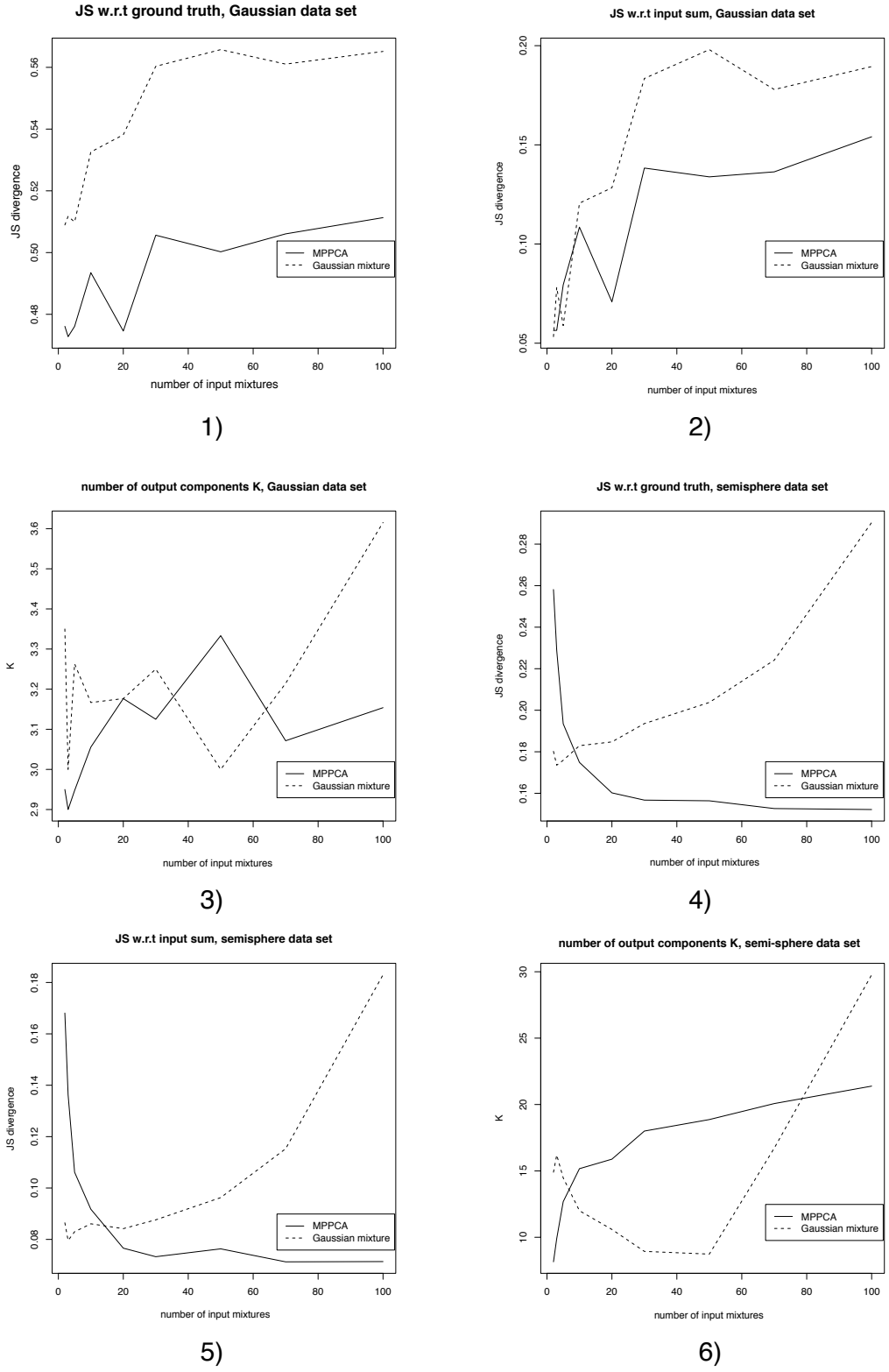


FIG. 4.4 – Evaluation de la qualité des agrégations avec les échantillons *Gaussian* et *semisphere*.

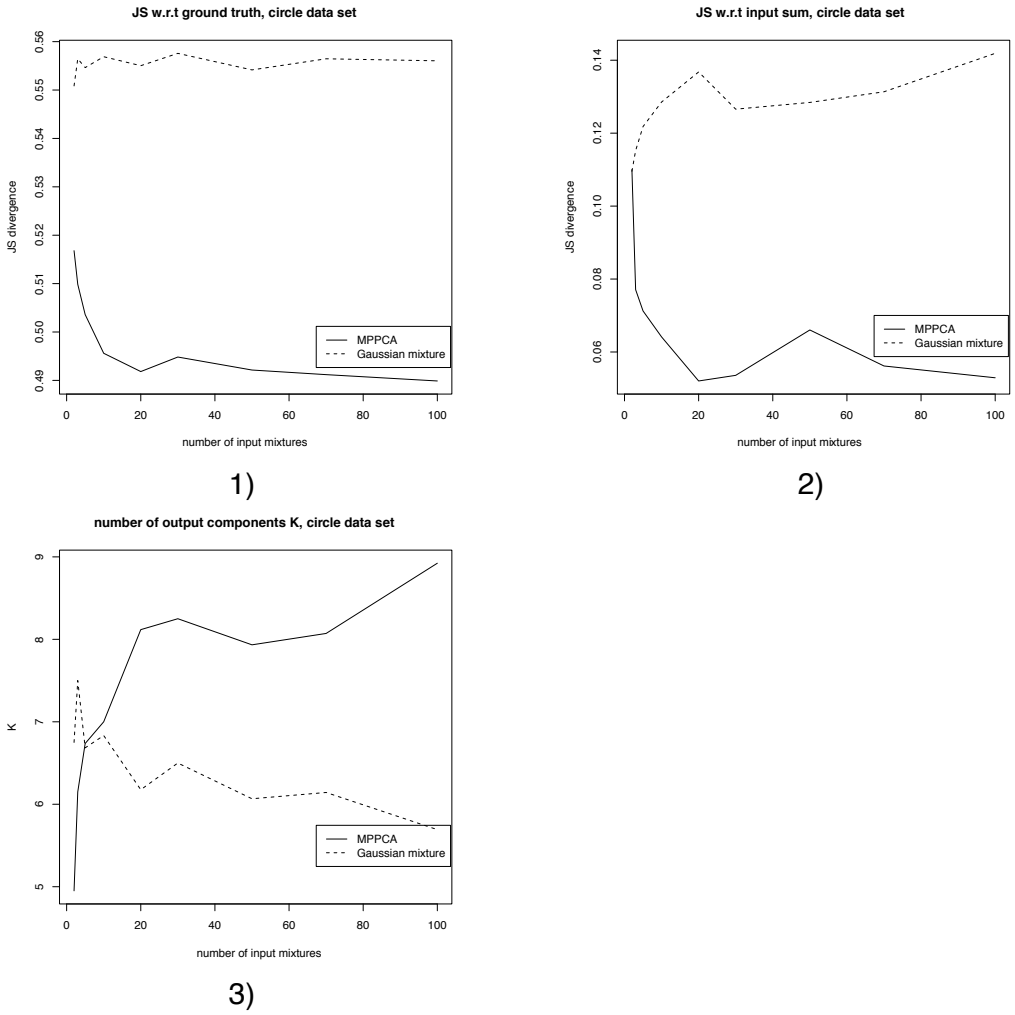


FIG. 4.5 – Evaluation de la qualité des agrégations avec l'échantillon *circle*.

Le rang des sous-espaces estimés est, en moyenne, conforme aux signaux originaux ; dans la table 4.1 on trouve une valeur de $q_{\text{sortie}} = 2.71$, alors que des gaussiennes 3D ont été utilisées. Les échantillons *circle* et *semi-sphere* présentent clairement des supports 1D ou 2D ; cette propriété est retrouvée expérimentalement (q_{sortie} est respectivement égal à 1.25 and 1.64 pour ceux-ci).

Sur les figures 4.4-(2,5) et 4.5-2, nous voyons que la perte de divergence KL (ou, de manière équivalente, JS) relativement aux modèles d'entrée est, en moyenne, plus élevée quand VBGMM-A est utilisé à la place de VBMPPCA-A. La seule exception survient quand un très petit nombre de sources est agrégé pour l'échantillon *semi-sphere* (figure 4.4-5).

Les mêmes remarques peuvent être faites quand nous utilisons le modèle estimé sur l'échantillon complet comme point de comparaison (figures 4.4-(1,4) et 4.5-1). Les méthodes variationnelles sont basées sur la minimisation d'une divergence KL (voir par. 3.5.1). En ce sens, VBMPPCA-A mène à des estimateurs plus précis.

Le nombre relatif de composantes produit par les méthodes VBGMM-A ou VBMPPCA-A semble dépendant de l'échantillon utilisé ; avec VBMPPCA-A, il est plus élevé pour *circle* (figure 4.5-3), et plus faible seulement quand un petit nombre de sources est utilisé pour *semi-sphere* (figure 4.4-6) et *Gaussian* (figure 4.4-3). Si cela illustre des comportements sensiblement différents, les complexités estimées par les deux méthodes ne sont pas incohérentes, et restent en général du même ordre.

4.6.3 Classification

Maintenant nous évaluons notre méthode sur un échantillon de données réelles. De manière analogue au paragraphe précédent, nous allons de nouveau comparer la qualité de nos modèles de sortie à une référence, mais nous allons également les confronter à une vérité terrain, matérialisée par une partition réelle.

En effet, les modèles de mélange sont souvent employés pour déduire une partition d'un échantillon de données ; chaque élément aura son étiquette inférée en utilisant la règle de décision de Bayes sur les composantes du mélange. Pour mesurer la qualité d'une partition inférée, nous utilisons la mesure d'erreur présentée dans la section 3.5.3.

Le protocole du précédent paragraphe a de nouveau été employé. 200 modèles d'entrée ont été estimés en utilisant VBMPPCA, sur des sous-échantillons plus grands (500 éléments). Les erreurs sont mesurées sur l'ensemble de l'échantillon, même pour les modèles d'entrée.

Les mesures relatives aux modèles d'entrée sont reportées dans la table 4.2, et les résultats d'expériences d'agrégation sont présentés sur la figure 4.6. Rappelons que l'échantillon utilisé dans cette section est défini sur 16 dimensions. Pour illustrer la capacité de VBMPPCA à limiter la charge de calcul, nous avons, cette fois-ci, imposé

$q_{\text{init}} = 8$, au lieu du réglage par défaut.

	K' (MPPCA)	K' (mélanges de gaussiennes)	q'
Pen data	15.97	23.95	6.35
	erreur de classification (%) (MPPCA)	erreur de classification (%) (mélanges de gaussiennes)	
Pen data	12.3	9.8	

TAB. 4.2 – Complexités obtenues (nombre de composantes K', dimensionalité moyenne des sous-espaces de chaque composante q') et erreurs de classification pour les modèles MPPCA d'entrée utilisés dans nos expériences. Les sous-échantillons utilisés pour créer ces modèles servent également à estimer des mélanges de gaussiennes avec VBGMM à des fins de comparaison.

En utilisant VBMPPCA sur nos sous-échantillons, en moyenne nous obtenons des modèles plus simples. Le nombre de composantes estimé est alors plus proche de la vérité terrain (16 pour MPPCA, 24 pour les mélanges de gaussiennes, contre 10 classes réelles, voir table 4.2). Nous voyons que même quand la taille des facteurs est contrainte a priori, en moyenne nous parvenons tout de même à réduire un peu plus les dimensionalités locales ($q_{\text{sortie}} = 6.35$).

L'erreur est significativement plus élevée en utilisant VBMPPCA, mais cela peut être dû à la forte différence de complexité des modèles estimés. En effet, l'augmentation du nombre de groupes réduit mécaniquement notre mesure d'erreur.

La divergence JS selon les modèles d'entrée d'agrégations réalisées avec VBMPPCA-A tend à être plus élevée, comme on peut remarquer dans la figure 4.6-2. La même constatation est valable quand on prend un modèle estimé sur la totalité de l'échantillon comme point de comparaison (voir fig. 4.6-1). Donc à première vue les résultats semblent moins bons avec VBMPPCA-A.

Cette dégénérescence peut venir de la limite que nous nous sommes imposés pour q_{init}. Toutefois, ce résultat doit être interprété avec discernement ; les données utilisées sont fortement non-gaussiennes, la borne inférieure optimisée comporte donc de très nombreux maxima locaux. Presque tous seront aussi bons du point de vue de la divergence KL optimisée, mais la divergence entre deux de ces maxima pourra être assez élevée.

Malgré ce handicap apparent, le nombre de groupes estimé avec VBMPPCA-A semble plus adéquat au regard de la vérité terrain (voir fig 4.6-3) ; VBGMM-A tend à fortement sous-estimer le nombre de composantes quand les sources de données deviennent nombreuses, alors que la complexité estimée semble plus stable avec VBMPPCA-A.

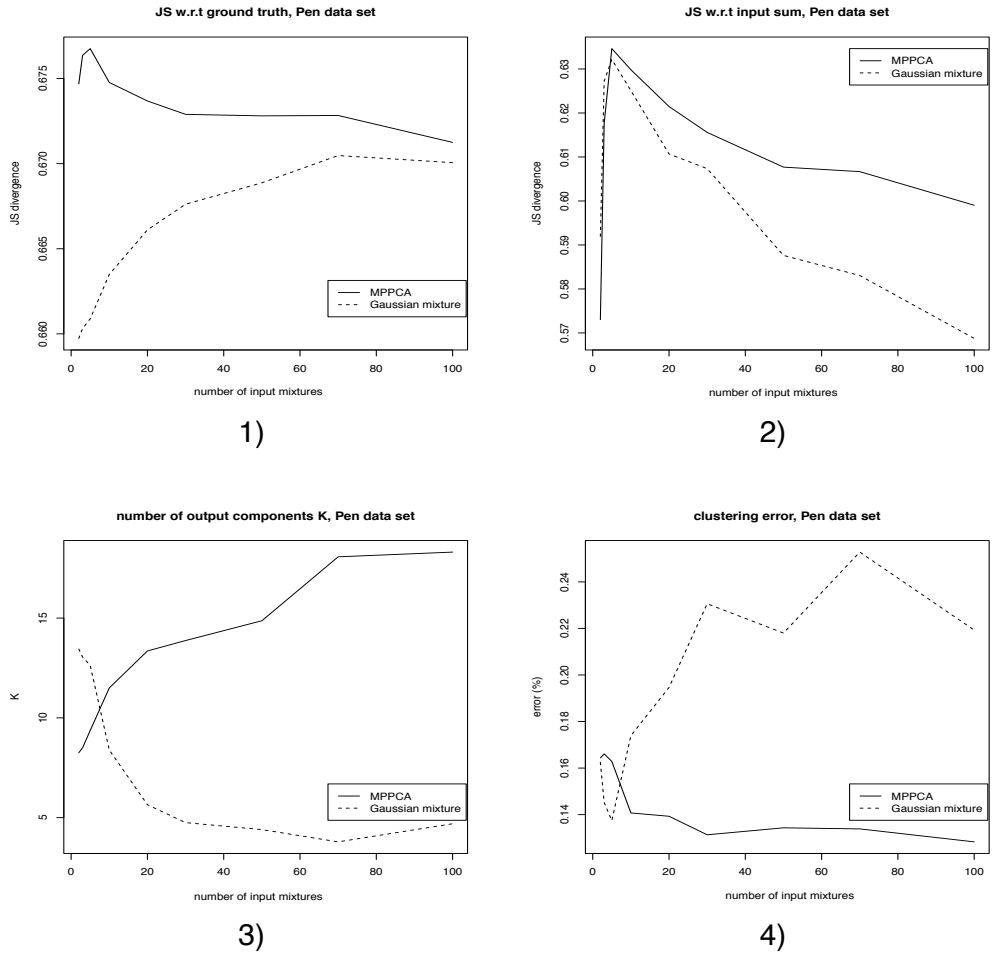


FIG. 4.6 – Evaluation de la qualité des agrégations avec *Pen data*.

En fait, les MPPCA d'entrée et les MPPCA agrégés ont des complexités similaires, alors qu'en utilisant des mélanges de gaussiennes on tend à obtenir des modèles d'entrée trop complexes et des agrégations trop simples. C'est sans doute une propriété intrinsèque de l'agrégation des MPPCA, qui utilise efficacement les combinaisons de sous-espaces pour compenser le bruit induit par des modèles d'entrée estimés sur des données limitées et les problèmes de minima locaux évoqués précédemment.

D'autre part, l'erreur observée sur des agrégations réalisées par VBMPPCA-A tend à être plus faible (voir fig. 4.6-4). Même si cet avantage doit être modéré par la nuance soulevée plus haut (i.e. les modèles plus complexes, ont, mécaniquement, une erreur plus faible), nous voyons que globalement VBMPPCA-A permet de mieux conserver l'information portée par les modèles d'entrée, même lorsqu'on utilise des matrices de facteurs de taille contrainte.

4.7 Aggrégation semi-supervisée de mélanges de gaussiennes

Cette section reprend essentiellement quelques unes de nos contributions [Bruneau et al., 2009a, 2010d].

4.7.1 Motivation

Considérons de nouveau le dispositif de fusion de mélanges de gaussiennes présenté dans la section 4.3. Chacun des dépôts de données est la source d'un mélange de gaussiennes ajusté sur les données locales. Dans la section 4.3, toutes les composantes de tous les modèles dont on dispose sont regroupées, formant ainsi un grand mélange, puis réduites par l'algorithme VBGMM-A.

Cependant, cette approche a un effet indésirable : la somme pondérée en entrée de l'algorithme devient alors un modèle extrêmement bruité, ce qui a pour conséquence une réduction drastique du nombre de composantes en sortie (voir section 4.7.6). Pour y remédier, nous allons décourager les regroupements de composantes d'un même modèle source vers une seule et même composante cible.

En ce sens, nous proposons une variante semi-supervisée (ou contrainte) de VBGMM-A. Dans le cas présent, la supervision n'est pas initiée par un utilisateur, mais par la structure de répartition des données en dépôts.

4.7.2 Modification de VBGMM-A

Nous proposons tout d'abord une modification de notre modèle probabiliste, de manière à prendre en compte l'origine de chaque composante à fusionner. Nous donnons ensuite l'algorithme permettant de calculer les solutions associées.

On considère que les L composantes en entrée sont associées à P sources distinctes (donc nécessairement $L \geq P$). On note a_{lp} la variable binaire indiquant si la composante l provient de la source p. On définit alors $\mathbf{A}$ la matrice à L lignes et P colonnes formée par les a_{lp}. L'origine de chaque composante est connue a priori, $\mathbf{A}$ est donc un jeu de données observé.

Nous allons alors associer une distribution de probabilité à ce nouveau jeu de données. Cette distribution évaluera à quel point les affectations estimées par l'algorithme variationnel violent ou valident les contraintes définies par $\mathbf{A}$. Restreignons donc la dépendance de $\mathbf{A}$ à la variable $\mathbf{Z}$. Autrement dit, on peut considérer $\mathbf{A}$ comme un échantillon d'une distribution paramétrée par $\mathbf{Z}$; dans ce contexte, un ensemble d'affectations (matérialisé par $\mathbf{Z}$) respectant les contraintes aura une plus grande vraisemblance.

Avant d'introduire la distribution en question, on considère la matrice $P \times K$ $\mathbf{M} = \mathbf{A}^T\mathbf{Z}$. Le terme m_{pk} mesure le nombre de composantes provenant de la source p associées à la composante cible k.

Compte tenu de la discussion précédente, on voudrait que ce nombre soit le plus petit possible, donc on choisit de modéliser nos contraintes avec une distribution de Poisson ayant pour paramètre $\lambda = 1$ sur chacun des termes de la matrice $\mathbf{M}$. Un tel paramétrage favorise les événements rares. Formellement, la distribution de probabilité associée à $\mathbf{A}$ est définie comme suit :

$$p(\mathbf{A}|\mathbf{Z}) = p(\mathbf{M} = \mathbf{A}^T\mathbf{Z}) = \prod_{p=1}^{P}\prod_{k=1}^{K} \frac{e^{-1}}{(1+m_{pk})!} \tag{4.39}$$

Le terme 1 dans la factorielle est ajouté pour la régularité des calculs à venir et ne cause pas de perte de généralité. On obtient une nouvelle distribution jointe pour le modèle en incorporant l'équation (4.39) au modèle de vraisemblance présenté dans la section 4.3.

Les algorithmes EM variationnels sont basés sur deux éléments essentiels : la mise à jour cyclique d'estimateurs inter-dépendants (cf application de la formule (3.41)) et l'augmentation monotone d'une borne inférieure à la vraisemblance marginale d'un échantillon $p(\mathbf{X})$.

Notre modification de modèle ne concerne que la variable $\mathbf{Z}$ aussi par la suite nous ne mentionnons que les termes l'impliquant, soit l'étape E de l'algorithme VBGMM-A.

Nous appliquons maintenant la formule (3.41) à notre modèle avec contraintes, et nous obtenons ainsi la distribution variationnelle optimale associée à $\mathbf{Z}$:

$$\begin{aligned}
\ln q^*(\mathbf{Z}) &= \mathbb{E}_{\omega,\mu,\Lambda}[\ln p(\mathbf{A},\mathbf{X},\mathbf{Z},\omega,\mu,\Lambda)] + \text{const} \\
\ln q^*(\mathbf{Z}) &= \mathbb{E}_{\omega}[\ln p(\mathbf{Z}|\omega)] + \mathbb{E}_{\mu,\Lambda}[\ln p(\mathbf{X}|\mathbf{Z},\mu,\Lambda)] + \ln p(\mathbf{A}|\mathbf{Z}) + \text{const}
\end{aligned} \tag{4.40}$$

On développe en utilisant les équations (4.8), (4.9) et (4.39), obtenant ainsi :

$$\begin{aligned}
\ln q^*(\mathbf{Z}) =& N\sum_{l=1}^{L}\sum_{k=1}^{K} z_{lk}\omega'_l\Bigg[\ln\tilde{\omega}_k + \frac{1}{2}\ln\tilde{\Lambda}_k - \frac{d}{2}\ln(2\pi) \\
&- \frac{1}{2}\left(\frac{d}{\beta_k} + \nu_k\mathrm{Tr}(\mathbf{W}_k\Lambda'^{-1}_l) + \nu_k(\mu'_l - \mathbf{m}_k)^T\mathbf{W}_k(\mu'_l - \mathbf{m}_k)\right)\Bigg] \\
&- \sum_{k=1}^{K}\sum_{p=1}^{P}\ln(1+m_{pk})! + \text{const}
\end{aligned}$$

Ou de manière équivalente :

$$\ln q^*(\mathbf{Z}) = \sum_{l=1}^{L}\sum_{k=1}^{K} z_{lk} \ln \rho_{lk} - \sum_{k=1}^{K}\sum_{p=1}^{P}\sum_{i=0}^{m_{pk}} \ln(1+i) + \text{const} \tag{4.41}$$

avec

$$\begin{aligned}
\ln \rho_{lk} = N\omega_l' \Bigg[& \ln \tilde{\omega}_k + \frac{1}{2} \ln \tilde{\Lambda}_k - \frac{d}{2} \ln(2\pi) \\
& - \frac{1}{2} \left(\frac{d}{\beta_k} + \nu_k \mathrm{Tr}(\mathbf{W}_k \Lambda_l'^{-1}) + \nu_k (\mu_l' - \mathbf{m}_k)^T \mathbf{W}_k (\mu_l' - \mathbf{m}_k) \right) \Bigg] \\
\tilde{\omega}_k = \mathbb{E}[\ln \omega_k] \text{ , et } & \tilde{\Lambda}_k = \mathbb{E}[\ln \det \Lambda_k]
\end{aligned}$$

On note $\mathbf{z}_{.k}$ l'ensemble $\{z_{lk} | \ \forall l\}$.

Habituellement, on peut factoriser $\ln q^*(\mathbf{Z})$ suivant l et k, donnant ainsi lieu à des estimateurs optimaux z_{lk} indépendants. Ici cela n'est plus valable, car les termes présents dans un ensemble $\mathbf{z}_{.k}$ sont co-dépendants : il faut faire évoluer l'étape E de notre algorithme.

On choisit un ordre arbitraire pour notre ensemble d'individus (i.e. de composantes en entrée), et on approche nos estimateurs co-dépendants en faisant une passe selon l'ordre choisi, et en utilisant les estimateurs découverts au fur et à mesure. Autrement dit, on fait l'approximation suivante :

$$q(\mathbf{Z}) = q(\mathbf{z}_1) q(\mathbf{z}_2|\mathbf{z}_1) q(\mathbf{z}_3|\mathbf{z}_1, \mathbf{z}_2) \dots q(\mathbf{z}_L|\mathbf{z}_1, \dots, \mathbf{z}_{L-1}) \tag{4.42}$$

Notre étape E est donc modifiée en ce sens. Nous allons décrire le traitement des deux premiers individus, et nous généraliserons. Ce procédé itératif conditionnel est assez proche d'ICM (iterated conditional modes) [Basu et al., 2006].

4.7.3 Initialisation du processus

On rappelle que $m_{pk} = \sum_{l=1}^{L} a_{lp} z_{lk}$. Le principe évoqué précédemment (voir équation (4.42)) nous permet, à chaque étape du traitement, de restreindre la somme au rang associé à cette étape. Pour la première étape nous avons donc :

$$\ln q^*(\mathbf{z}_1) = \sum_{k=1}^{K} z_{1k} \ln \rho_{1k} - \sum_{k=1}^{K}\sum_{p=1}^{P} \ln(1 + a_{1p} z_{1k}) + \text{const} \tag{4.43}$$

Ce qui, pour un k particulier, donne :

$$\ln q^*(z_{1k}) = z_{1k} \ln \rho_{1k} - \sum_{p=1}^{P} \ln(1 + a_{1p} z_{1k}) + \text{const} \tag{4.44}$$

On voit alors difficilement comment se ramener à un estimateur de loi multinomiale. Toutefois, en utilisant un développement du premier ordre de $\ln(1+x)$ selon la formule de Taylor, nous obtenons :

$$\ln q^*(z_{1k}) = z_{1k} \ln \rho_{1k} - \sum_{p=1}^{P} a_{1p} z_{1k} + \text{const} \tag{4.45}$$

$$\ln q^*(z_{1k}) = z_{1k} \ln \frac{\rho_{1k}}{e^{\sum_{p=1}^{P} a_{1p}}} + \text{const} \tag{4.46}$$

Étant donné que chaque composante n'appartient qu'à une seule source,

$$\ln q^*(z_{1k}) = z_{1k} \ln \frac{\rho_{1k}}{e} + \text{const} \tag{4.47}$$

Ce qui nous donne un estimateur non-normalisé $\rho'_{1k} = \frac{\rho_{1k}}{e}$. Dans ce cas de figure, on retombe logiquement sur le même estimateur normalisé que lorsque les contraintes ne sont pas prises en compte.

4.7.4 Une nouvelle formule générale de mise à jour pour $q^*(\mathbf{Z})$

On passe à l'étape suivante de notre traitement, ce qui revient à une restriction au rang 2 de l'équation (4.43). En particulier, on a alors :

$$\ln q^*(\mathbf{z}_2|\mathbf{z}_1) = \sum_{k=1}^{K} z_{2k} \ln \rho_{2k} - \sum_{k=1}^{K}\sum_{p=1}^{P} \ln(1 + a_{1p} z_{1k}) - \sum_{k=1}^{K}\sum_{p=1}^{P} \ln(1 + a_{1p} z_{1k} + a_{2p} z_{2k}) + \text{ c} \tag{4.48}$$

De nouveau, on considère un k particulier, et le développement de Taylor associé, menant alors à :

$$\ln q^*(z_{2k}|z_{1k}) = z_{2k} \ln \rho_{2k} - \sum_{p=1}^{P} a_{1p} z_{1k} - \sum_{p=1}^{P} (a_{1p} z_{1k} + a_{2p} z_{2k}) + \text{ c} \tag{4.49}$$

Nous remarquons qu'à ce stade, z_{1k} est déjà connu, nous nous intéresserons donc à la valeur relative de z_{2k}. Pour ce faire, on note $a_{i\max} = \arg\max_p a_{ip}$ et $z_{i\max} = \arg\max_k z_{ik}$. Compte tenu de ces considérations, l'expression peut être arrangée comme suit :

$$\ln q^*(z_{2k}|z_{1k}) = z_{2k}(\ln \rho_{2k} - 1 - 2\delta_{a_{1\max}, a_{2\max}} . \delta_{z_{1\max}, k}) + \text{ c} \tag{4.50}$$

avec δ l'opérateur de Kronecker. On aboutit alors à l'opérateur dénormalisé suivant :

$$\rho'_{2k} = \frac{\rho_{lk}}{e^{1+2\delta_{a_{1\max}, a_{2\max}} . \delta_{z_{1\max}, k}}}$$

Pour un rang quelconque, un raisonnement similaire mène à la formule générale suivante :

$$\rho'_{jk} = \frac{\rho_{jk}}{e^{1+\sum_{i=1}^{j-1}(j-i+1).\delta_{a_{i\max},a_{j\max}}.\delta_{z_{i\max},k}}} \tag{4.51}$$

où j est le rang de l'individu en cours de traitement.

L'algorithme de réduction de mélanges de gaussiennes incluant cette étape E modifiée sera dénommé VBGMM-B par la suite.

4.7.5 Modification de la borne inférieure

La borne inférieure évoquée précédemment est une somme d'espérances mathématiques suivant les distributions variationnelles courantes. Par souci de concision, nous nous concentrons sur les modifications induites par l'étape E modifiée.

Les expressions de borne inférieure pour les algorithmes VBGMM et VBGMM-A peuvent être trouvées respectivement dans [Bishop, 2006, chap. 10] et la section 4.3. Ici nous la résumerons simplement par la variable $\mathcal{F}$. L'inclusion de contraintes a modifié la distribution variationnelle de $\mathbf{Z}$. On a donc une nouvelle borne inférieure $\mathcal{F}'$:

$$\mathcal{F}' = \mathcal{F} + \mathbb{E}[\ln p(\mathbf{A}|\mathbf{Z})] \tag{4.52}$$

$$= \mathcal{F} + \sum_{k=1}^{K}\sum_{p=1}^{P}\left[-1 - \sum_{i=0}^{\mathbb{E}[m_{pk}]}\ln(1+i)\right] \tag{4.53}$$

$$= \mathcal{F} - KP - \sum_{k=1}^{K}\sum_{p=1}^{P}\sum_{i=0}^{\mathbb{E}[m_{pk}]}\ln(1+i) \tag{4.54}$$

avec

$$\mathbb{E}[m_{pk}] = \mathbb{E}\left[\sum_{l=1}^{L} a_{lp}z_{lk}\right] = \sum_{l=1}^{L} a_{lp}\mathbb{E}[z_{lk}] = \sum_{l=1}^{L} a_{lp}r_{lk} \tag{4.55}$$

Avec l'algorithme VBGMM-A, cette borne inférieure est assurée de suivre une croissance stricte. Dans le contexte présent, à des fins d'efficacité, nous avons mis en oeuvre une étape E heuristique, aussi cette propriété n'est plus garantie ; de légères décroissances peuvent être observées.

Mais l'implémentation associée ne change pas fondamentalement : on observe la valeur de la borne inférieure à chaque itération, et on utilise $\Delta(\text{borne inférieure}) < \text{seuil}$ comme critère d'arrêt, sauf que désormais Δ peut être négatif. Alternativement on peut mesurer l'agitation de $\mathbf{Z}$ (voir formule (3.68).

4.7.6 Evaluation expérimentale de l'algorithme VBGMM-B

Dans cette section, nous allons comparer VBGMM-B à VBGMM-A en utilisant une collection d'images. Pour cela, nous avons sélectionné les 10 premières catégories de la

collection d'images Caltech-256 [Griffin et al., 2007], formant ainsi un ensemble de 1243 images.

Chaque image est alors vue comme une source de données, et un mélange de gaussiennes est ajusté sur les intensités des pixels (espace colorimétrique (L,a,b), augmenté des positions de pixels (x,y)). Les mélanges individuels obtenus sont en moyenne formés de 18 composantes. Le protocole consiste alors à choisir aléatoirement x sources puis à réaliser l'agrégation avec une des méthodes évoquées plus haut. Nous mesurerons alors :

- la divergence KL de l'agrégat obtenu par rapport à la somme pondérée de composantes fournie en entrée, et par rapport à chaque source de données prise séparément,

- le nombre d'itérations avant convergence,

- le nombre de composantes finalement retenu pour le modèle cible.

Les résultats sont donnés dans les figures 4.7, 4.8 et 4.9 pour différentes valeurs de x. De ces résultats, nous tirons les conclusions suivantes :

- VBGMM-B donne des résultats équivalents à VBGMM-A du point de vue de la divergence KL. Un légère dégradation peut être parfois observée.

- Cette fois-ci, nous observons que la prise en compte de l'origine des composantes évite une réduction excessive du nombre de composantes quand le nombre de modèles en entrée devient grand : quand 100 sources sont agrégées (soit 1800 composantes en moyenne), on obtient 40 composantes au lieu de 9. Le processus sous-jacent n'est pas très clairement défini (i.e. les images d'une même catégorie peuvent présenter un profil de signal assez différent), aussi il est souvent vraisemblable de procéder à des fusions drastiques.

  Dans ce cas de figure, nos contraintes peuvent être interprétées comme une information fournie a priori, et le compromis se fera entre la préservation de cette information et une estimation de qualité. Il en résulte donc un nombre plus important, mais cependant raisonnable, de composantes.

- Quand le nombre de sources devient grand, VBGMM-B devient beaucoup plus rapide que VBGMM-A. Intuitivement, pour VBGMM-A, un temps considérable est passé à effectuer des regroupements intempestifs qui n'apportent que peu de gain au niveau de la divergence KL, et plus largement au niveau de la borne inférieure. En revanche, les contraintes permettent d'arrêter le processus avant que survienne cet artefact.

Commentons la complexité algorithmique relative des algorithmes : les itérations de VBGMM-A ont une complexité suivant $O(L)$, que l'inclusion de contraintes fait passer

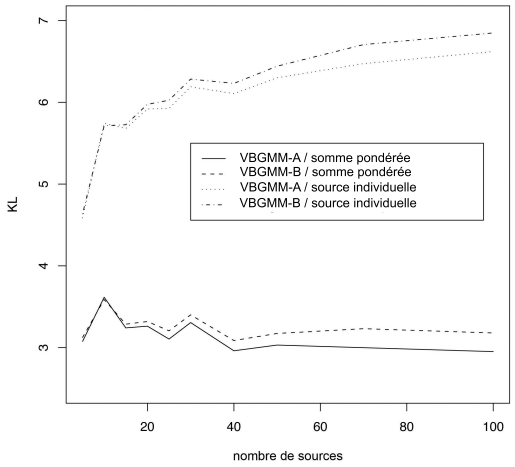


FIG. 4.7 – Divergence KL par rapport à la somme pondérée fournie en entrée pour VBGMM-A et VBGMM-B

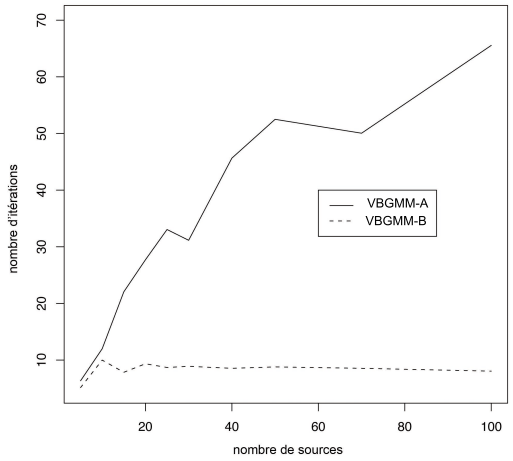


FIG. 4.8 – Nombre d'itérations avant convergence pour VBGMM-A et VBGMM-B

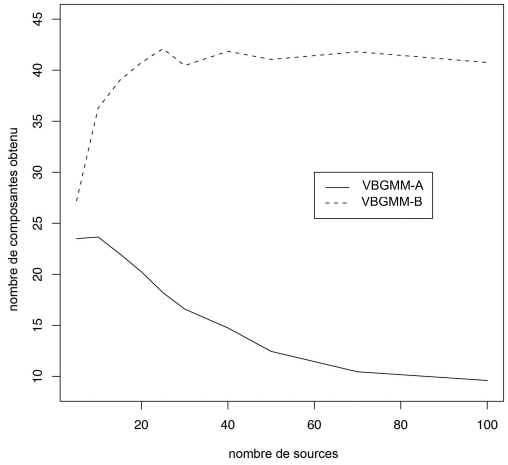


FIG. 4.9 – Nombre de composantes dans le modèle cible pour VBGMM-A et VBGMM-B

à $O(L\ln(L))$. Cependant, sur la figure 4.8, on remarque que le nombre d'itérations pour VBGMM-A semble linéaire selon L, alors que VBGMM-B semble en $O(1)$.

Donc quand L devient grand, le coût cédé par une itération serait largement compensé par le gain en nombre d'itérations. De plus, une implémentation élégante de l'étape E modifiée ramène la complexité de celle-ci à $O(L)$, ce qui en renforce d'autant plus l'intérêt.

4.8 Conclusion

Nous avons présenté nos contributions dans le domaine de l'agrégation de modèles de mélanges, en mettant l'accent sur la parcimonie des solutions trouvées et la rigueur des solutions algorithmiques déclinées. L'agrégation de MPPCA permet même d'envisager une parcimonie supérieure, en ajustant le rang de chaque composante aux données supportées. Nous avons illustré nos propositions par des expériences. Nous aurions pu employer des échantillons de dimensionalités plus élevées, mais nous avons justifié de notre choix dans un but de comparaison.

La prise en compte de l'origine de chaque composante (i.e. aspect semi-supervisé), outre les qualités algorithmiques en découlant, ouvre la voie à l'intégration de contraintes utilisateur. On peut en effet facilement imaginer qu'au lieu d'être imposées par la structure du réseau, l'utilisateur impose des contraintes en fonction de ses préférences, guidant ainsi le processus. Ce modèle pourrait par ailleurs tout aussi bien être employé avec l'algorithme VBGMM standard (voir section 3.5.2).

Chapitre 5

Classification visuelle par nuages d'agents

5.1 Introduction

Dans ce chapitre, nous proposons une méthode de classification non-supervisée et visuelle adaptée au contexte semi-supervisé et incrémental. Notre contribution est une variante de l'algorithme proposé dans [Picarougne et al., 2007]. Ce chapitre a fait l'objet de publications [Bruneau et al., 2008b, 2009b].

Notre approche, d'inspiration bio-mimétique, tranche singulièrement avec les chapitres 3 et 4. Nous commençons donc ce chapitre par un état de l'art spécifique à ce type de fonctions de classification. Ensuite, nous introduisons l'algorithme FClust [Picarougne et al., 2007], dont la contribution de ce chapitre est dérivée. Cette dernière consiste en deux opérateurs gérant la dimension semi-supervisée et incrémentale de l'algorithme, décrits par la suite. Enfin, des résultats expérimentaux illustrent la proposition.

5.2 Classification bio-mimétique

Dans les années 70, les travaux fondateurs sur les algorithmes évolutionnaires ont été concentrés sur les algorithmes génétiques (GA) [Holland, 1975], les stratégies d'évolution (ES) [Rechenberg, 1965] et la programmation évolutionnaire (EP) [Fogel et al., 1966]. Dans l'esprit, ces trois catégories d'algorithmes ont utilisé des principes similaires, car elles sont toutes inspirées du néo-Darwinisme : i.e. l'utilisation d'une population d'individus (e.g. dans le cas de la classification non-supervisée d'un ensemble d'éléments, une partition possible pour ceux-ci), l'évaluation des individus sur la base d'une fonction de coût ou de qualité, la sélection (ou survie) des individus les mieux adaptés, et la génération d'une nouvelle population avec des opérateurs génétiques comme le croisement et la mutation.

Pour toutes ces approches, les représentations initiales (i.e. individus codés de manière binaire, réelle, ou hiérarchique) et les principes évolutionnaires (opérateurs) contraignent la façon dont ces algorithmes sont appliqués au problème de la classification. Par exemple, les algorithmes génétiques reposent en partie sur l'usage de l'opérateur de croisement, réalisant l'échange de gènes entre individus ; aussi dans le cadre de la classification il faut définir comment deux partitions possibles d'un même échantillon peuvent

échanger des caractéristiques.

De manière analogue, comme ES utilise un codage des individus par des nombres réels, il faut trouver comment encoder une partition sous forme de vecteur réel, et comment en réaliser la mutation, par exemple avec des lois gaussiennes. Les algorithmes génétiques ont notamment été appliqués au problème de la classification en utilisant différents codages [Cucchiara, 1993, Von Laszewski, 1991, Jones and Beltramo, 1991, Falkenauer, 1994, Hall et al., 1999, Nasraoui and Leon, 2004]. Un individu représente une partition de l'échantillon de données, et une population de tels individus évolue selon des opérations de sélection, recombinaison et mutation, comme suggéré par la théorie de l'évolution de Darwin.

Quelques années plus tard, les fourmis artificielles ont inspiré les informaticiens dans de nombreux domaines. Cela est principalement dû au fait que le comportement de ces animaux est assez riche et diversifié. Dans les années 90, des biologistes ont modélisé le comportement de fourmis réelles, et simulé comment celles-ci pourraient partitionner, ou regrouper, des objets trouvés dans leur nid.

Deneubourg et al. apparaissent comme les pionniers de cette approche, et ont proposé un premier modèle pour le tri d'objets selon leur similarité par des fourmis artificielles [Deneubourg et al., 1990, Goss and Deneubourg, 1991] : plus précisément, les fourmis artificielles se déplacent alors sur une grille 2D. Les objets à trier sont placés de manière aléatoire sur la grille. Chaque fourmi est placée sur cette même grille, n'a qu'une perception locale de la cellule proche de sa position, et ne communique pas avec les autres fourmis. La configuration locale des objets sur la grille influence le comportement des fourmis. Le pas entre tri et classification d'objets a été franchi dans [Lumer and Faieta, 1994, Azzag et al., 2006b, Kuntz et al., 1997, Monmarché et al., 1999, Azzag et al., 2006a] avec des adaptations de l'algorithme présenté ci-dessus. Différentes approches biologiques ont été proposées pour orienter le déplacement des fourmis : la reconnaissance chimique dans [Labroche et al., 2002], ou l'auto-assemblage dans [Azzag et al., 2004].

L'algorithme du réseau immunitaire présente un autre exemple d'adaptation à la classification ; les anti-gènes représentent les données, et les anti-corps représentent les partitions qui doivent être adaptés aux anti-gènes [De Castro and Von Zuben, 2000, Timmis et al., 2000, Nasraoui et al., 2003].

Un avantage de ces algorithmes inspirés de comportements biologiques est leur nature intrinsèquement distribuée, sans contrôle central, impliquant ainsi des coûts de communication faibles. La recherche effectuée est souvent plus globale relativement à des approches standard. De plus, aucune connaissance a priori n'est en général requise (e.g. partition initiale ou nombre de groupes final), à l'opposé de l'algorithme K-means [McQueen, 1967], par exemple.

L'intelligence d'essaim regroupe les algorithmes basés sur une population d'agents élémentaires [Bonabeau et al., 1999]. Les fourmis artificielles peuvent être associées à cette classe, mais dans ce chapitre nous nous concentrons sur les modèles de mouvement d'un nuage d'agents ou de particules appliqués à la résolution d'un problème d'optimisation. Par exemple, la méthode PSO (*Particle Swarm Optimization*, ou optimisation par essaim de particules [Clerc, 2005]) utilise une population de particules placées dans un espace de recherche, et représentant des solutions possibles à un problème d'optimisation. Ces particules sont caractérisées par une position et une vitesse, ainsi que par la possibilité d'interagir les unes avec les autres ; elles peuvent ainsi coordonner localement leurs mouvements, afin de trouver le maximum de la fonction.

Ces principes ont été déjà été appliqués au problème de la classification [Proctor and Winter, 1998]. Chaque agent est associé à un élément de données. Il peut réagir à la présence d'autres agents dans son voisinage, en prenant en compte sa similarité avec ceux-ci. De cette façon, les agents se déplacent vers une position où ils sont les plus proches d'éléments qui leurs sont similaires.

En appliquant une telle règle de comportement, les agents forment progressivement des groupes de données similaires se déplaçant de manière coordonnée. Une des caractéristiques intéressantes de cette classe d'algorithmes est la possibilité de produire une visualisation indépendante de la nature exacte des données traitées (notamment de leur dimensionalité). La position et la vitesse des agents fournit une information visuelle sur les groupes découverts. Ces algorithmes sont ainsi des candidats intéressants dans l'optique d'un partitionnement interactif.

Le présent chapitre se base sur une évolution de l'algorithme FClust [Picarougne et al., 2007]. Celui-ci est inspiré d'un modèle biologique du comportement social des animaux en mouvement (e.g. nuages d'oiseaux) [Reynolds, 1987]. Le modèle proposé place les individus dans un environnement 2D ou 3D, qui vont ensuite se déplacer suivant des règles de décision. Ces règles sont en général les mêmes pour tous les individus, et ne prennent en compte que le voisinage de chacun, i.e. les autres individus dans un périmètre donné. L'application de ces règles détermine l'amplitude et la direction du mouvement de chaque individu. Par exemple, un individu peut s'éloigner de ses voisins, ou se rapprocher des individus qui lui sont similaires, etc. Résultant de ces adaptations locales, des formes complexes peuvent apparaitre dans le nuage d'agents, une propriété souvent nommée *émergence* dans la littérature de biologie comportementale.

Les mouvements d'individus artificiels peuvent être utilisés dans le contexte d'une fonction de classification : chaque individu se déplaçant dans l'environnement représente un élément de l'échantillon de données, et essaiera de se déplacer de manière à être proche des individus qui lui sont similaires. Ainsi, après une évolution suffisamment longue, des groupes d'individus similaires se déplaceront ensemble, suggérant naturellement une partition des données.

Grâce à l'abstraction d'éléments complexes par un environnement 2D ou 3D, cette

méthode permet une visualisation directe du processus de classification, se rapprochant alors du domaine de la fouille de données visuelle [Cleveland, 1993, Keim and Kriegel, 1996, Wong and Bergeron, 1997]. Les méthodes Isomap [Tenenbaum et al., 2000], LLE [Roweis and Saul, 2000], ou SOM (*Self-Organizing Map*) [Kohonen, 1990] permettent d'extraire un support topologique des données afin de les représenter sur un espace à faible dimension. FClust diffère en ce que l'espace de représentation n'est pas explicitement relié à une information topologique, mais sert plutôt de terrain pour que les mouvement des agents leur permettent de s'organiser spontanément. De ce fait, la représentation utilisée par FClust est intimement liée à l'algorithme de classification sous-jacent. Pour contraster, dans le chapitre 6, nous présenterons une méthode de visualisation de classification où technique de visualisation et algorithme d'apprentissage sont beaucoup plus décorrélés.

5.3 L'algorithme FClust

5.3.1 Intuition

Dans cette section, nous rappelons l'algorithme FClust [Picarougne et al., 2007], le fondement de la contribution de ce chapitre. Celui-ci est basé sur la formation et le mouvement de nuages d'agents dans un espace clos. Une illustration de ce principe est donnée dans la figure 5.1. Partant d'une situation initiale (a) où les agents sont dispersés au hasard dans l'environnement (avec des directions aléatoires), nous souhaitons obtenir la situation finale (b) dans laquelle les agents similaires bougent de manière cohérente (dans la même direction, et proches les uns des autres).

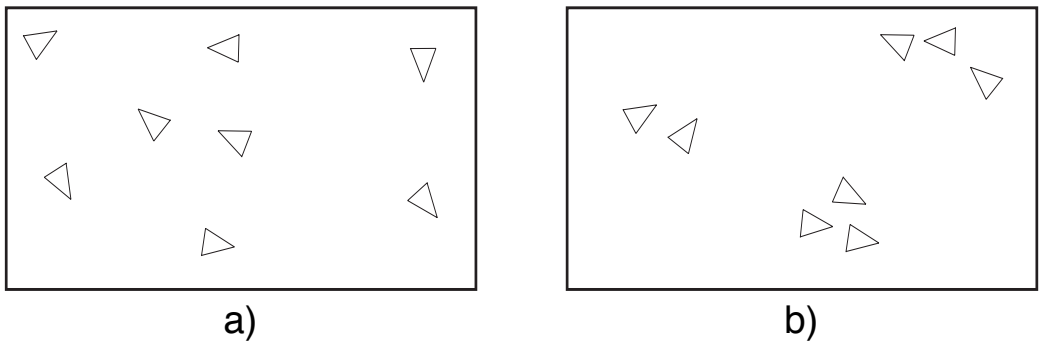


FIG. 5.1 – Illustration de l'environnement dans lequel évolue un nuage d'agents. Partant d'une situation initiale (a) où les agents étaient placés aléatoirement, on souhaite arriver à la situation (b), où les agents similaires se déplacent de manière cohérente (extrait de [Picarougne et al., 2007]).

5.3.2 Présentation

L'approche FClust est basée sur le positionnement d'agents (chaque agent étant associé à un élément de données) dans un espace 2D ou 3D, puis l'application de simples règles de décision, inspirées de celles appliquées dans les formations d'oiseaux ou les bancs de poissons : pour chaque agent, se rapprocher des agents similaires ; s'éloigner des agents dissimilaires. Ces règles sont telles, qu'après une évolution suffisamment longue (ou, dans un contexte informatique, après suffisamment d'itérations) des groupes d'agents (ou, de manière équivalente, de données) homogènes sont créés.

Considérons un ensemble de N éléments $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ à partitionner. La seule hypothèse qui est formée sur les données est l'existence d'une mesure de similarité sur leur espace, notée $\text{sim}(i, j)$.

L'algorithme 5 pour contrôler les agents fonctionne comme suit : initialement, tous les agents sont placés dans des positions et directions aléatoires. Les distances idéales sont alors calculées pour chaque couple d'agents. Ensuite les agents bougent et décident, selon une règle locale, s'ils devraient se rapprocher et/ou aller dans la même direction (et réciproquement).

Intuitivement, cette règle a deux buts : 1) faire converger les distances effectives entre agents vers leur distance idéale, et 2) faire en sorte que les agents associés à des données similaires se déplacent dans la même direction. De cette règle locale émergent des groupes d'agents qui se déplacent ensemble, définissant ainsi une partition des données traitées.

Entrées : N éléments $\mathbf{x}_1, \ldots, \mathbf{x}_N$
Sorties : Partition des éléments
1 **Placer** (aléatoirement, par exemple) les N agents dans l'environnement 2D ;
2 **tant que** *pas de convergence* **faire**
3 | **Calculer** nouvelle position pour chacun des N agents selon même règle locale ;
4 | **Déplacer** tous les agents simultanément ;
5 **fin**
6 **Construire** les groupes depuis le nuage d'agents ;

Algorithme 5 : Première version de l'algorithme FClust pour le contrôle des agents

Le déplacement de tous les agents est déterminé par la même règle locale. Nous n'entendons pas ici en faire une description exhaustive, mais plutôt intuitive. Pour plus de détails, le lecteur peut consulter [Picarougne et al., 2007]. Pour chaque agent i, à chaque itération, cette règle calcule un nouveau vecteur de vitesse dépendant à la fois de ses position et vitesse actuelles, ainsi que du comportement et de la nature des agents dans son voisinage. Plus précisément, seuls les agents situés à une distance inférieure à un seuil vont influencer i. Cette distance est estimée selon la nature des données.

Si le voisinage de l'agent i est vide, celui-ci continue son déplacement sans changer de direction ou de vitesse. Si un ou plusieurs agents sont présents dans le voisinage de i, leur influence peut être décomposée en 2 parties : en termes de direction et d'amplitude.

5.3.3 Changement de direction

Dans l'optique d'une illustration claire, considérons dans un premier temps le cas où un seul agent j est présent dans le voisinage de i. L'influence de j sur i en termes de direction dépend à la fois de $d(i, j)$ relativement à une distance idéale (notée $d^*(i, j)$), et de l'angle entre les vecteurs de vitesse de j et de i.

La distance idéale est calculée selon les similarités entre les éléments de données $\mathbf{x}_i$ et $\mathbf{x}_j$. La règle de déplacement est conçue de telle sorte que le mouvement de deux agents relativement similaires tend à s'aligner, ceux-ci étant alors séparés par leur distance idéale (voir figure 5.2).

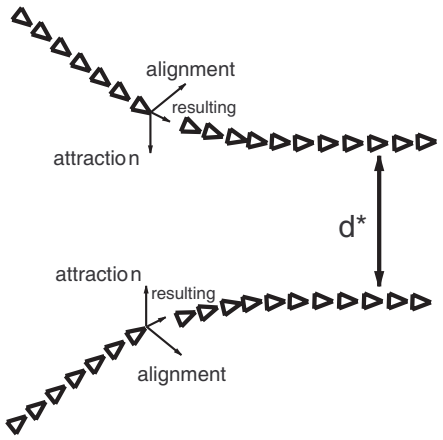


FIG. 5.2 – Comportement théorique de deux agents en interaction (extrait de [Picarougne et al., 2007])

Plus précisément :

- L'attraction ou la répulsion entre deux agents est calculée relativement à leurs distances actuelle et idéale,
- L'alignement dépend de l'angle actuel entre leurs vecteurs de vitesse.

Nous traitons le cas où plusieurs agents sont présents dans le voisinage de i simplement en sommant puis normalisant leur influence.

5.3.4 Changement d'amplitude et convergence

Le vecteur de vitesse de i est mis à jour selon le nombre d'agents présents dans son voisinage, de sorte que :

- aucun agent ne reste seul et immobile dans l'environnement sans la possibilité de rencontrer d'autres agents,
- les groupes d'agents aient tendance à se déplacer de moins en moins vite en grossissant,
- et l'on assure une vitesse limitée à l'agent isolé, de manière à ne pas le placer directement au centre d'un groupe. En effet, on pourrait ainsi provoquer l'éclatement du groupe, s'il s'avère que l'agent est assez dissimilaire d'une majorité de ses membres.

Nous obtenons alors le résultat recherché, i.e. l'agent est progressivement rattaché au groupe adéquat. Les agents isolés auront plus tendance à se déplacer d'un groupe à l'autre afin de trouver celui qui leur convient le mieux.

L'algorithme utilise des pas de temps pour réaliser les mouvements, et tous les agents se déplacent de manière synchrone, mettant à jour leur directions et amplitudes à chaque pas de temps. Les frontières de l'environnement se comportent comme des miroirs : quand un agent atteint la frontière, il rebondit et part dans la direction symétrique selon la normale de la frontière.

5.4 Adaptation semi-supervisée et incrémentale

FClust ne peut pas être utilisé dans le contexte d'un flux de données, car le nombre d'agents exploserait avec la taille de l'échantillon traité, rendant la visualisation encombrée. Pour pallier cet inconvénient, l'utilisation d'une fenêtre glissante a été proposée dans la littérature [Lavergne et al., 2007].

Nous choisissons une stratégie différente, en permettant la formation de résumés d'information par fusion d'agents. Plus précisément, afin d'éviter d'avoir à stocker toutes les données reçues, nous proposons de représenter un groupe d'agents fusionnés par une densité de probabilité gaussienne multivariée paramétrée par (μ, Σ). Pour appliquer ce principe, le calcul des similarités de FClust doit être adapté.

Initialement, chaque agent i est associé à une distribution avec pour moyenne $\mathbf{x}_i$, et une covariance faible. Quand le nombre d'agents présents dans la simulation devient trop grand, les fusions procèdent à partir de cette situation initiale. Nous avons choisi d'utiliser la distance Euclidienne entre le centre des distributions comme critère de décision des agents à fusionner.

Nous introduisons deux opérateurs pouvant être appliqués à ces agents : un opérateur de fusion (*Merge*) et un opérateur d'éclatement (*Split*). L'opérateur de fusion nous permet de réduire le nombre d'agents présents dans la simulation. Grossièrement, cet

opérateur prend 2 groupes d'agents (autrement dit 2 gaussiennes) et construit un nouvel agent en fusionnant ces distributions. Cela permet d'effectuer la classification de notre flux de données en temps linéaire selon le nombre d'élements traités.

L'opérateur d'éclatement permet d'adapter notre partition à des changements structurels dans le flux de données reçu. Nous considérons un contexte semi-supervisé pour notre algorithme (pour une définition, voir la section 2.4) : ainsi certains éléments du flux sont associés avec des classes a priori. Si l'opérateur de fusion décide de regrouper 2 agents associés à des éléments de plus d'une modalité d'étiquette, l'opérateur d'éclatement permettra de diviser ce dernier afin de ne récupérer une population d'agents associés qu'à au plus une étiquette.

5.4.1 Opérateur de fusion

Nous proposons un opérateur de fusion dans une optique incrémentale. Ainsi, nous ne stockons que μ (l'espérance des données des agents groupés), Σ (la matrice de covariance associée) et π (le nombre d'éléments de données groupés) pour chaque agent.

Cet opérateur prend 2 agents $(\pi_{k1}, \mu_{k1}, \Sigma_{k1})$ et $(\pi_{k2}, \mu_{k2}, \Sigma_{k2})$ en entrée et génère un nouvel agent (π_k, μ_k, Σ_k). La matrice de covariance du nouvel agent peut être calculée incrémentalement en utilisant une généralisation de la formule de décomposition de la covariance d'une gaussienne $\Sigma = \mathbb{E}[\mathbf{X}.\mathbf{X}^T] - \mu.\mu^T$.

L'équation (5.1) montre comment construire un nouvel agent k à partir de 2 agents $k1$ et $k2$.

$$\begin{aligned}
\pi_k &= \pi_{k1} + \pi_{k2} \\
\mu_k &= \frac{\pi_{k1}.\mu_{k1} + \pi_{k2}.\mu_{k2}}{\pi_{k1} + \pi_{k2}} \\
\Sigma_k &= \mathbb{E}_k[\mathbf{X}_k.\mathbf{X}_k^T] - \mu_k.\mu_k^T \\
&= \left(\frac{\mathbb{E}_{k1}[\mathbf{X}_{k1}.\mathbf{X}_{k1}^T].\pi_{k1}}{\pi_{k1} + \pi_{k2}} + \frac{\mathbb{E}_{k2}[\mathbf{X}_{k2}.\mathbf{X}_{k2}^T].\pi_{k2}}{\pi_{k1} + \pi_{k2}} \right) - \mu_k.\mu_k^T \\
&= \frac{(\Sigma_{k1} + \mu_{k1}.\mu_{k1}^T).\pi_{k1} + (\Sigma_{k2} + \mu_{k2}.\mu_{k2}^T).\pi_{k2}}{\pi_{k1} + \pi_{k2}} - \mu_k.\mu_k^T
\end{aligned} \tag{5.1}$$

Nous voulons utiliser cet opérateur quand trop d'agents sont présents dans la simulation à un temps t, ceci afin de réduire la complexité de l'algorithme. Pour que l'algorithme de classification conserve de bonnes propriétés, il nous faut également ne fusionner que des agents similaires, menant ainsi à des distributions gaussiennes plus compactes.

Les trois conditions suivantes sont donc définies pour décider de l'usage de l'opérateur de fusion sur deux agents :

– Il y a plus de $idealNbA$ agents dans la simulation.

– Un agent a au moins un autre agent dans son voisinage.

– La similarité entre deux agents est plus grande qu'un seuil M_{th}.

La fusion provoque ainsi une décroissance du nombre d'agents nbA dans la simulation. La première condition permet de réguler le nombre d'agents dans la simulation, tout en empêchant des fusions intempestives : si trop peu d'agents évoluent dans l'espace, la probabilité de rencontre entre deux agents devient très faible. La probabilité de former des groupes est alors également faible, d'où une baisse de la vitesse de convergence de l'algorithme. La troisième condition permet d'assurer que des agents suffisamment similaires sont fusionnés. Le seuil M_{th} est défini dans l'équation (5.2), avec nbA le nombre d'agents actuellement dans la simulation, et $simM$ la similarité moyenne entre tous les nbA agents.

$$M_{th} = \begin{cases} 1 & \text{si } (nb1 < idealNbA) \\ \frac{1-simM}{nbA-simM} + simM & \text{sinon} \end{cases} \tag{5.2}$$

5.4.2 Opérateur d'éclatement

L'opérateur d'éclatement nous permet de générer deux agents à partir d'un seul. Nous créons ainsi deux gaussiennes multivariées $(\pi_{k1}, \mu_{k1}, \Sigma_{k1})$ et $(\pi_{k2}, \mu_{k2}, \Sigma_{k2})$ à partir d'une composante initiale (π_k, μ_k, Σ_k).

Cet opérateur est choisi de manière à vérifier l'inverse de l'opérateur de fusion. Nous avons choisi l'éclater la composante gaussienne selon un vecteur propre de la matrice de covariance associée, et de placer les moyennes des 2 nouvelles distributions en fonction de la valeur propre associée. Cette méthode a déjà été appliquée dans [Blekas and Lagaris, 2007].

Comme nous travaillons dans un contexte semi-supervisé, nous utilisons les données étiquetées pour décider du choix du vecteur propre utilisé. L'équation (5.3) montre comment réaliser l'éclatement d'un agent k en deux nouveaux agents $k1$ et $k2$ avec λ_i et v_i la valeur propre et le vecteur propre de la matrice Λ_k choisis.

$$
\begin{aligned}
\pi_{k1} &= \pi_{k2} \\
&= \frac{\pi_k}{2} \\
\mu_{k1} &= \mu_k + \frac{\sqrt{\lambda_i}}{2}.v_i \\
\mu_{k2} &= \mu_k - \frac{\sqrt{\lambda_i}}{2}.v_i \\
\Sigma_{k1} &= \Sigma_{k2} \\
&= \Sigma_k + \mu_k.\mu_k^T - \frac{\mu_{k1}.\mu_{k1}^T + \mu_{k2}.\mu_{k2}^T}{2} \\
&= \Sigma_k - \frac{\lambda_i}{4}.v_i.v_i^T
\end{aligned} \tag{5.3}
$$

Nous utilisons cet opérateur si, après une opération de fusion, des données étiquetées avec au moins 2 modalités de classe différentes apparaissent dans cette classe. Nous remarquons qu'en appliquant cet opérateur juste après avoir appliqué l'opérateur de fusion, nous ne pouvons pas avoir plus de 2 modalités de classe a priori différentes dans une composante qui vient d'être fusionnée.

L'éclatement n'a donc besoin d'être fait qu'en deux composantes. Les 5 étapes suivantes définissent le processus d'éclatement :

- extraire les données étiquetées des paramètres de l'agent (π_k, μ_k, Σ_k),
- calculer les centroides p_1 et p_2 des deux groupes de données g_1 et g_2 ayant une étiquette différente,
- prendre le vecteur propre v_i le plus colinéaire avec le vecteur formé par p_1 et p_2, et la valeur propre associée λ_i de Σ_k,
- calculer les nouvelles composantes $(\pi_{k1}, \mu_{k1}, \Sigma_{k1})$ et $(\pi_{k2}, \mu_{k2}, \Sigma_{k2})$ en appliquant l'opérateur d'éclatement sur (π_k, μ_k, Σ_k) avec v_i et λ_i,
- associer les données étiquetées du groupe g_i avec $(\pi_{ki}, \mu_{ki}, \Sigma_{ki})$.

À la première étape, nous extrayons les données étiquetées de l'agent avant de réaliser l'éclatement, afin que ces valeurs ne soient pas oubliées. Par conséquent, si les données à partitionner sont résumées par des opérations de fusions successives, on s'assure bien que les données étiquetées par un utilisateur restent explicitement mémorisées ; celles-ci ne seront en général pas très nombreuses, le surplus de stockage ne sera donc pas significatif.

Ensuite, l'opération revient à éclater la distribution selon une droite qui divise au

mieux les 2 groupes de données étiquetées (i.e $\min |\cos(\overrightarrow{p_1p_2}, v_i)|$).

Afin de limiter le temps de calcul nécessaire à la détermination du vecteur propre, nous ne prendrons en compte que les trois vecteur propres associés aux 3 valeurs propres maximales. Ces valeurs peuvent être efficacement approchées en utilisant l'algorithme *power method*.

5.4.3 Algorithme général

Nous inscrivons nos contributions dans l'algorithme général suivant :

Entrées : Flux de données
Sorties : Position, amplitude et direction d'un population de NbA agents
tant que *il reste éléments dans flux* **faire**
  Lire éléments dans flux ;
  Créer agents pour éléments lus ;
  tant que $NbA > idealNbA$ **faire**
    Réaliser fusions d'agents ;
    si $\exists$ *2 étiquettes* $\neq$ *dans le même agent* **alors**
      Réaliser éclatement ;
    fin
  fin
  Calculer nouvelles vitesses et positions d'agents ;
  Déplacer agents ;
fin

Algorithme 6 : Algorithme incrémental et semi-supervisé

5.4.4 Fonction de classification finale

Quand toutes les données ont été traitées, nous déterminons les groupes de manière analogue à la méthode FClust. Afin de réaliser une évaluation expérimentale, nous devons associer chaque élément du flux d'origine, dont ceux qui ont été fusionnés, à un groupe spécifique. Cela est réalisé simplement en associant chaque élément d'origine i à l'agent présent dans la simulation dont la moyenne associée μ est la plus proche de $\mathbf{x}_i$. Nous avons par ailleurs conditionné la taille du voisinage d'un agent au nombre et à la nature des données qui lui sont associées : en particulier, le paramètre M_{th} associé à chaque agent est adapté au moyen d'une fenêtre glissante. En quelque sorte, nous combinons notre stratégie avec celle utilisée dans [Lavergne et al., 2007].

5.5 Interactions utilisateur

Une des propriétés intéressantes de cet algorithme est de permettre une visualisation dynamique du processus de classification. Cette visualisation est possible sur l'espace de simulation (i.e. 2D ou 3D) sans condition préalable sur le nombre de dimensions des données traitées.

Cela permet également de gérer des interactions de la part d'un utilisateur. Cette propriété a déjà été évoquée dans [Picarougne et al., 2007].

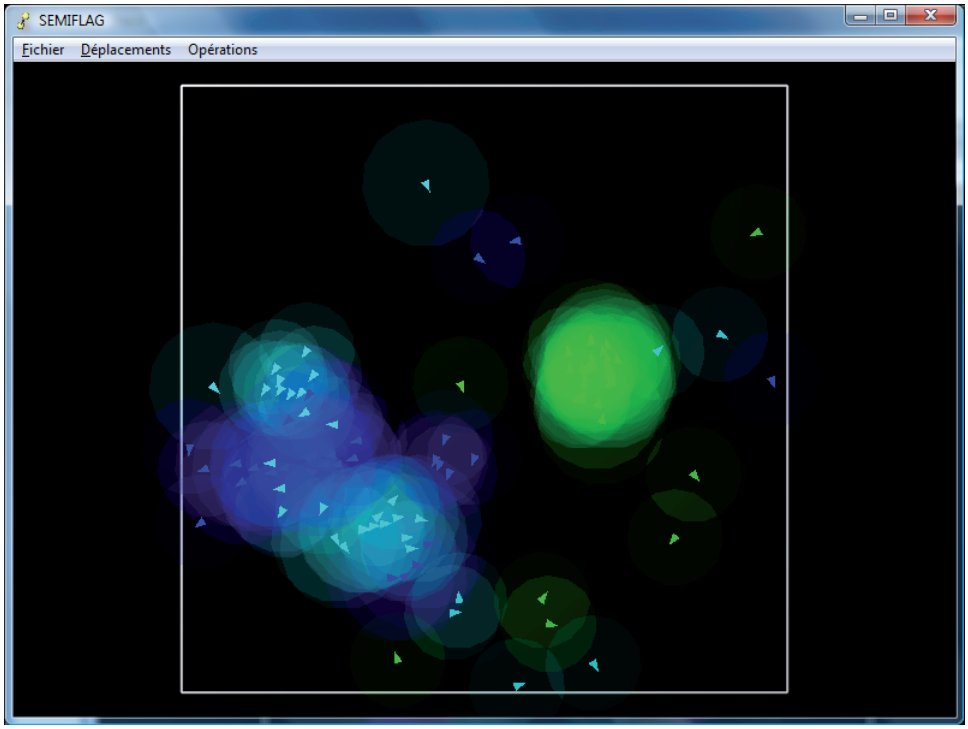


FIG. 5.3 – Visualisation des groupes d'individus avec FClust

La figure 5.3 montre une capture d'écran de notre application. Nous représentons un agent par un triangle, et la taille de son voisinage par un halo circulaire autour de lui. Celle-ci est fonction du nombre d'éléments que l'agent résume. L'utilisateur peut, s'il le souhaite, figer temporairement l'affichage afin de pouvoir visualiser les caractéristiques d'un agent particulier.

À l'opposé de l'algorithme FClust, nous ne pouvons pas afficher toutes les données associées avec un agent car l'information est résumée dans les composantes gaussiennes. Néanmoins, l'utilisateur peut accéder aux données étiquetées. L'utilisateur peut aussi demander au système quel est l'agent le plus proche d'un élément de données particulier (au sens de la similarité). D'autre part, nous avons considéré les éléments pré-étiquetés comme intrinsèques du flux traité ; nous pourrions aussi envisager de permettre à l'utilisateur d'étiqueter des éléments via l'interface de visualisation.

5.6 Résultats expérimentaux

Afin d'évaluer notre nouvel algorithme, nous l'avons testé sur plusieurs jeux de données réels (UCI [Asuncion and Newman, 2007]) et synthétiques (définis dans [Picarougne et al., 2007, Lavergne et al., 2007]). Ces jeux de données ont été sélectionnés car les véritables classes de leurs éléments sont connues (voir figure 5.4).

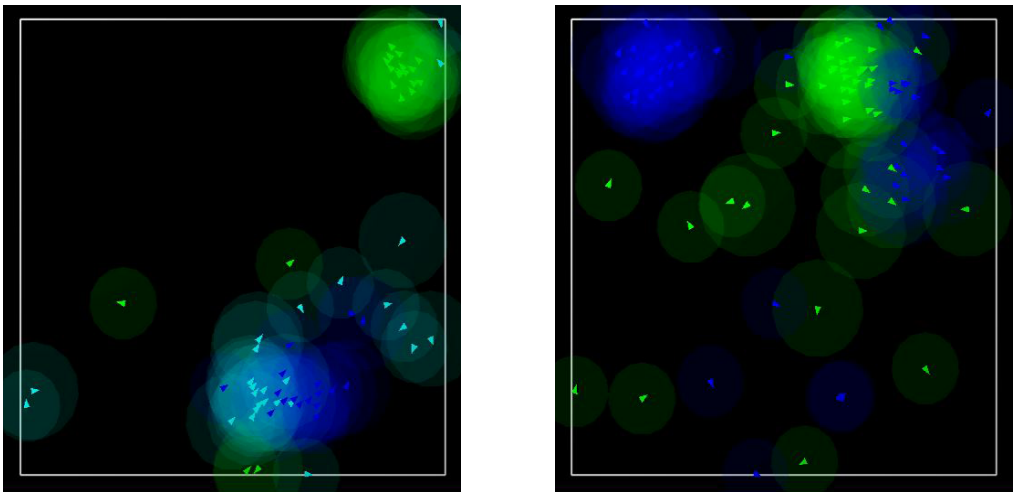


FIG. 5.4 – Instantané de notre méthode sur les données iris (à g.) et ART2 (à d.). Les couleurs des agents dénotent les classes réelles, présentées à titre indicatif mais non utilisées par l'algorithme.

Dans un premier temps nous avons étudié l'influence du paramètre $idealNbA$, et évalué notre méthode en comparaison de FClust et de la variante incrémentale proposé dans [Lavergne et al., 2007]. Comme ces méthodes n'utilisent pas d'information semi-supervisée, nous avons d'abord comparé ces trois algorithmes sans utiliser l'opérateur d'éclatement défini précédemment. Pour les mêmes raisons, nous ne prenons pas en compte l'interaction utilisateur dans ces tests. Les classes réelles des éléments ne sont pas utilisées pour l'exécution des méthodes, mais plutôt afin d'évaluer les partitions calculées.

La table 5.1 montre les résultats obtenus. Art2 et Art3 sont des données artificielles obtenues par génération depuis diverses lois gaussiennes (classes avec recouvrement, attributs ne portant pas d'information ou bruités, etc). Les données 10000-ta et 25000-ta sont obtenues par le mélange de 5 lois uniformes, et nous servent à illustrer l'utilisation d'échantillons de grande taille. Nous voyons que notre algorithme incrémental obtient sensiblement les mêmes résultats que l'algorithme classique (voir [Picarougne et al., 2007]), mais avec une complexité linéaire selon le nombre d'éléments traités.

Données	$n_{\text{population}}$ ($n_{\text{cl. réel}}$)	idealNbA=50 $n_{\text{cl. trouvé}}$	pureté	idealNbA=100 $n_{\text{cl. trouvé}}$	pureté
PIMA	768 *(2)*	4.70 *[1.42]*	0.69 *[0.02]*	2.20 *[1.32]*	0.67 *[0.02]*
waveform	5000 *(3)*	3.50 *[1.43]*	0.58 *[0.12]*	3.60 *[1.51]*	0.57 *[0.06]*
ART2	1000 *(2)*	4.70 *[1.57]*	0.94 *[0.07]*	3.70 *[1.34]*	0.98 *[0.01]*
ART3	1100 *(4)*	6.80 *[3.01]*	0.89 *[0.01]*	2.70 *[0.95]*	0.81 *[0.17]*
10000-ta	10000 *(5)*	6.50 *[1.27]*	1.00 *[0.00]*	5.20 *[0.45]*	1.00 *[0.00]*
25000-ta	25000 *(5)*	6.10 *[1.45]*	1.00 *[0.00]*	5.00 *[0.00]*	1.00 *[0.00]*
Données	$n_{\text{population}}$ ($n_{\text{cl. réel}}$)	idealNbA=150 $n_{\text{cl. trouvé}}$	pureté		
PIMA	768 *(2)*	3.00 *[0.67]*	0.69 *[0.02]*		
waveform	5000 *(3)*	2.90 *[1.10]*	0.54 *[0.08]*		
ART2	1000 *(2)*	2.30 *[0.95]*	0.88 *[0.20]*		
ART3	1100 *(4)*	2.30 *[1.25]*	0.80 *[0.18]*		
10000-ta	10000 *(5)*	5.17 *[0.41]*	1.00 *[0.00]*		
25000-ta	25000 *(5)*	5.20 *[0.45]*	1.00 *[0.00]*		

TAB. 5.1 – Résultats obtenus (nombre de classes réel, trouvé, et pureté des partitions) avec notre algorithme incrémental pour 3 valeurs du paramètre *idealNbA*. Chaque résultat est calculé selon une moyenne sur 10 exécutions (l'écart-type est indiqué entre crochets)

En comparaison de l'autre variante incrémentale de FClust (voir [Lavergne et al., 2007]), nous obtenons un nombre de groupes beaucoup plus proche du nombre de classes réel, avec un temps de calcul comparable (les algorithmes ont tous deux une complexité linéaire). Dans ces résultats, nous remarquons que la densité d'agents dans la simulation est un paramètre très sensible : avec trop peu d'agents on obtient trop de classes, et trop d'agents provoquent une chute de la performance en termes de complexité et de qualité des partitions obtenues.

Une valeur de $idealNbA$ à 100 semble un bon compromis entre nombre de groupes et qualité de la partition. Nous utilisons cette valeur afin de tester les autres paramètres.

Dans la table 5.2 nous avons testé l'efficacité de notre algorithme en mesurant le temps passé à partitionner divers jeux de données comportant un nombre variable d'éléments. Nous constatons empiriquement la complexité linéaire de notre algorithme. Remarquons que celle-ci est grossièrement quadratique selon $idealNbA$.

Notre algorithme peut être appliqué à de grandes bases de données : avec $idealNbA = 100$, nous traitons 10000 éléments en 70.92 ± 0.91s et 25000 éléments en 174.52 ± 2.30s sur un processeur Xeon dual core 3.2GHz avec 4Gb de mémoire.

Finalement, nous avons testé l'influence de l'usage de l'opérateur d'éclatement. Afin d'évaluer son efficacité, nous avons choisi de faire varier la proportion du flux de données étiqueté (*propData*). *propData* doit être relativement petit dans le contexte

Données	$n_{\text{population}}$	idealNbA=50	idealNbA=100	idealNbA=150
PIMA	768	0.85 *[0.04]*	3.25 *[0.34]*	6.83 *[0.64]*
waveform	5000	6.63 *[0.15]*	27.06 *[0.80]*	60.73 *[0.88]*
ART2	1000	1.83 *[0.10]*	7.01 *[0.19]*	16.43 *[1.76]*
ART3	1100	1.90 *[0.08]*	7.21 *[0.62]*	16.54 *[0.84]*
10000-ta	10000	19.99 *[0.27]*	70.92 *[0.91]*	159.13 *[2.08]*
25000-ta	25000	49.50 *[0.35]*	174.52 *[2.30]*	397.46 *[2.27]*

TAB. 5.2 – Temps d'exécution de notre algorithme pour 3 valeurs du paramètre *idealNbA*. Chaque résultat est calculé selon une moyenne sur 10 exécutions (l'écart-type est indiqué entre crochets)

semi-supervisé.

Données	$n_{\text{population}}$ ($n_{\text{cl. réel}}$)	propData=0.01		propData=0.05	
		$n_{\text{cl. trouvé}}$	pureté	$n_{\text{cl. trouvé}}$	pureté
PIMA	768 *(2)*	1.6 *[0.89]*	0.65 *[0.00]*	1.6 *[0.89]*	0.65 *[0.00]*
waveform	5000 *(3)*	2.60 *[0.89]*	0.55 *[0.07]*	2.60 *[0.55]*	0.56 *[0.04]*
ART2	1000 *(2)*	5.0 *[1.22]*	0.97 *[0.01]*	4.80 *[1.64]*	0.98 *[0.01]*
ART3	1100 *(4)*	3,67 *[0.58]*	0.88 *[0.02]*	5.80 *[0.45]*	0.88 *[0.01]*
Données	$n_{\text{population}}$ ($n_{\text{cl. réel}}$)	propData=0.1			
		$n_{\text{cl. trouvé}}$	pureté		
PIMA	768 *(2)*	1.6 *[0.89]*	0.66 *[0.01]*		
waveform	5000 *(3)*	2.60 *[0.55]*	0.55 *[0.05]*		
ART2	1000 *(2)*	5.00 *[1.00]*	0.97 *[0.02]*		
ART3	1100 *(4)*	5.80 *[1.92]*	0.88 *[0.01]*		

TAB. 5.3 – Influence de la proportion (*propData*) de données supervisées dans le flux de données , avec *idealNbA* fixé à 100. Chaque résultat est calculé selon une moyenne sur 10 exécutions (l'écart-type est indiqué entre crochets)

La table 5.3 résume les résultats obtenus avec 3 valeurs de ce paramètre (0.01, 0.05 et 0.1). Nous observons que l'usage de cet opérateur nous permet d'obtenir des résultats légèrement meilleurs en termes de nombre de groupes découverts, ou de pureté des classes, sur la moitié des jeux de données testés. Mais la différence est très marginale selon la valeur de *propData* utilisée. Cela montre que l'influence de notre opérateur d'éclatement est limitée.

5.7 Conclusion

Dans ce chapitre, nous avons décrit une nouvelle méthode qui étend l'algorithme FClust, permettant de créer et visualiser dynamiquement des groupes de données dans un échantillon. Plus précisément, nous avons proposé une adaptation semi-supervisée et incrémentale de l'algorithme. Les résultats obtenus sont similaires à ceux obtenus avec

FClust, mais avec une complexité de calcul linéaire selon la taille de l'échantillon. Notre approche incrémentale mène à de meilleurs résultats que d'autres tentatives d'adaptation, en particulier au regard du nombre de groupes découverts. Toutefois, notre opérateur d'éclatement, constituant la dimension semi-supervisée de notre contribution, n'apporte qu'une amélioration marginale des résultats. Cet aspect est donc une piste d'amélioration majeure.

Chapitre 6

Visualisation de classification d'une collection d'images

6.1 Introduction

Dans ce chapitre, nous allons présenter une méthode originale pour la visualisation de classification d'une collection d'images. Notre système est obtenu en combinant plusieurs modules, dont certains sont inspirés de nos contributions des chapitres précédents. Notre technique d'agrégation de mélanges de gaussiennes (voir section 4.3) est en effet utilisée pour former les prototypes de groupes d'images. Nous proposons une fenêtre glissante (voir section 2.5) pour gérer un flux de données. Une interface 3D originale, utilisant les similarités entre prototypes, et autorisant l'interaction avec un utilisateur, est détaillée.

Ce chapitre a fait l'objet de contributions personnelles [Bruneau et al., 2010e,f].

Le contexte de ce travail est d'abord exposé : nous justifions notamment de la combinaison de techniques de classification et de visualisation, relativement à l'existant du domaine. La représentation des images, et la technique de partitionnement de celles-ci, sont ensuite spécifiées. Nous exposons également une adaptation incrémentale de l'algorithme, permettant de traiter les images à la volée. Ce processus sert de tâche de fond au système de visualisation présenté subséquemment. Ce dernier utilise une représentation de graphe en 3D, exploitant les similarités entre prototypes des groupes d'images. Un algorithme classique force et ressort simule un système physique, intuitif pour un être humain, capable de prendre en compte la nature dynamique des similarités calculées par un partitionnement incrémental. L'interface autorise l'utilisateur à rétro-agir sur le système, influençant ainsi l'algorithme sous-jacent de classification de la collection d'objets.

6.2 Présentation du système

L'extraction basée sur le contenu d'une collection d'images est un sujet d'importance dans la littérature [Datta et al., 2005]. Nous considérons des collections d'images de thèmes arbitraires, comme ce qui pourrait être récupéré sur le Web ou partagé par des utilisateurs de Flickr. Dans ce contexte, plusieurs but distincts peuvent être recherchés. D'un côté, on pourrait chercher l'apparition d'un objet ou d'un type de scène spécifique, ayant alors à résoudre des problèmes de variabilité d'apparence ou d'occlusion [Mikolajczyk and Schmid, 2001]. Dans ce cas, un objectif et une vérité terrain bien

clairs existent. Un autre champ d'application important concerne le respect des droits de propriété sur les contenus [Poullot et al., 2007], tâche qui peut être partiellement automatisée.

Plus récemment, des auteurs ont étudié la caractérisation de classes dans une collection d'images [Carbonetto et al., 2008]. Par exemple, dans certaines situations l'utilisateur pourrait initialement ne pas avoir clairement identifié un objet recherché, et l'utilisateur est alors inclus dans la boucle de recherche, soit en notant la pertinence des résultats retournés afin de guider l'algorithme (i.e. *relevance feedback*), soit en proposant un mode de navigation interactive.

6.2.1 Principes de visualisation

Dans le présent chapitre, nous entendons poursuivre l'objectif de navigation interactive, en proposant un outil convivial pour facilement observer et interagir avec une collection d'images potentiellement en perpétuelle évolution. Nous décrivons une méthode de partitionnement pour images de manière à offrir une vue synthétique à un utilisateur. Une représentation dynamique sous forme de graphe lui permet d'interagir facilement avec la partition proposée. Ainsi nous proposons un moyen efficace et intuitif d'explorer une collection d'images.

La nouveauté de la contribution ne se situe pas au niveau des caractéristiques extraites des images, qui seront exposées dans la section expérimentale. Nous avons choisi de décrire chaque image par un mélange de modèles probabilistes estimé sur l'ensemble de ses pixels, abstractions qui seront ensuite utilisées pour l'algorithme de partitionnement. De ce postulat, nous déclinons l'approche suivante :

- la partition a priori de la collection d'images en groupes visuellement similaires,
- la visualisation de cette structure grâce à un graphe 3D dont les noeuds et les arcs reflètent les images supportées et les similarités entre groupes,
- l'adaptation du partitionnement et de la visualisation à un contexte incrémental, afin de gérer une collection d'images pouvant évoluer à tout moment,
- la possibilité pour l'utilisateur de rétro-agir sur la partition en manipulant la visualisation.

Afin de réduire la charge cognitive de l'utilisateur, au démarrage du système, celui-ci n'est confronté qu'à l'évolution des similarités entre groupes. Pour ce faire, un graphe complet, dont les noeuds représentent les groupes, est construit. Pour informer l'utilisateur sur le contenu de chaque noeud, et en même temps limiter le contenu présent à l'écran, chaque noeud n'expose que quelques miniatures de ses images les plus représentatives. Les longueurs d'arcs sont directement reliées aux similarités entre

groupes. Pendant le processus, le nombre de groupes et leurs similarités relatives sont amenés à évoluer ; le graphe représentatif est alors mis à jour en utilisant un algorithme *force et ressort* [Kamada and Kawai, 1989, Fruchterman and Reingold, 1991].

La visualisation d'un graphe complet peut être pénible, et peu efficace avec un algorithme force et ressort. En effet, les ressorts dominent alors la répulsion entre les noeuds. Aussi, considérant un nombre de groupes possible borné (maximum 20), nous permettons à l'utilisateur de régler un seuil de distance pour le filtrage d'arcs, ajustable sous condition que le graphe ne reste formé que d'une seule composante connexe. L'utilisateur peut aussi interagir avec la visualisation en inspectant le contenu exhaustif d'un groupe en particulier, en imprimant une rotation au graphe afin de le voir sous un autre angle, en modifiant la géométrie du graphe en déplaçant des noeuds, ou en agissant sur l'algorithme en proposant des suppressions ou créations de groupes.

L'essentiel des travaux existants traite l'interaction utilisateur sous la forme d'une réponse à une requête, ou à un *univers clos* (i.e. classes connues à priori). Par exemple, dans [Carson et al., 2002], les auteurs modélisent des segments de texture et de couleur avec un mélange de gaussiennes. Les utilisateurs forment une requête en choisissant un ou plusieurs segments depuis une image spécifique, et les résultats pertinents sont extraits de la collection.

Le système présenté dans [Chen et al., 2003] est plus proche de l'approche présente, avec la construction de groupes locaux au voisinage d'une image présentée en requête par l'utilisateur. Dans [Vailaya et al., 2001], une méthode de classification exploitant une représentation hiérarchique est proposée. Les classes sont pré-déterminées, et l'apprentissage des caractéristiques les plus discriminantes est effectuée de manière supervisée.

6.2.2 Cadre pour la classification

Nous proposons un système de visualisation du partitionnement de la collection d'images, ce qui revient à découvrir de l'information et de la structure au sein de données a priori non structurées et sans étiquettes particulières. Le partitionnement est réalisé sur des mélanges de gaussiennes (i.e. chaque image est représentée par un mélange de gaussiennes, et ce dernier est un élément dans le cadre d'un algorithme de classification), et à notre connaissance, le partitionnement de telles densités de probabilité n'a pas été traité dans la littérature.

Dans notre approche, l'interaction utilisateur n'affine pas une réponse à une requête, mais capture la préférence de l'utilisateur vis-à-vis de l'algorithme de classification. Cela revient à mettre en oeuvre une méthode semi-supervisée. Seuls quelques exemples de telles méthodes sont donnés ici, pour un état de l'art plus complet, voir [Chapelle et al., 2006], ou encore notre section dédiée 2.4.

Nigam et al. [Nigam et al., 2000] utilisent quelques étiquettes fournies par un utilisateur pour initialiser un partitionnement, et partant de cette initialisation, estiment ensuite un modèle de mélange avec un algorithme EM. Basu et al. [Basu et al., 2004] modélisent un ensemble de contraintes avec des champs de Markov cachés (*hidden Markov random fields*), et utilisent ce formalisme pour proposer un algorithme K-means avec une fonction objectif modifiée en conséquence. Une étape d'adaptation de cette fonction est également incluse.

6.3 Partitionnement d'une collection d'images

Cette tâche doit être décomposée en deux phases successives, partiellement indépendantes : choisir une représentation adéquate pour les images, et décliner une technique de partitionnement utilisant ces représentations. Nous commençons par introduire une représentation spatiale et colorimétrique jointe pour les pixels de nos images. Les pixels d'une image sont ensuite modélisés par un mélange de gaussiennes estimé par l'algorithme VBGMM 3.5.2. Ceci nous permet notamment d'obtenir des modèles avec un nombre adéquat de composantes. Nous proposons ensuite un algorithme itératif de partitionnement des représentations ainsi construites. Nous proposons un algorithme inspiré de K-means [McQueen, 1967, Lloyd, 1982].

6.3.1 Représentation d'image

Plutôt que le classique espace colorimétrique (R,G,B), nous utiliserons (L,a,b), ce dernier étant connu pour ses bonnes propriétés perceptuelles, menant ainsi à des mesures de distance plus significatives [Wyszecki and Stiles, 1982, Goldberger et al., 2003]. Pour transcrire l'organisation spatiale de l'image, nous ajouterons les coordonnées de pixels à notre espace de description. Nous travaillerons ainsi sur l'espace à 5 dimensions (L,a,b,x,y) pour décrire les pixels de nos images. Dans le cas où les images à traiter ne seraient pas toutes de taille identique, les positions des pixels peuvent être normalisées sur $[0, 1]$ sans perte de généralité.

Réaliser le partitionnement d'une collection d'images en utilisant des dissimilarités à l'échelle des pixels n'est pas réalisable en pratique. Nous adoptons une représentation plus compacte, en estimant un mélange de gaussiennes sur l'ensemble de pixels d'une image, décrits sur notre espace (L,a,b,x,y). Nous réduisons ainsi la taille des données traitées, en regroupant les pixels selon leur homogénéité. La construction de mesures de similarité sur de telles représentations a déjà été étudiée dans la littérature [Goldberger et al., 2003, Vasconcelos, 2001]. Nous utilisons l'algorithme VBGMM pour construire ces représentations synthétiques.

6.3.2 Partitionnement itératif de mélanges de gaussiennes

Construction des centroides approchés

Dans la section précédente, nous avons décrit la caractérisation utilisée pour les images, et le choix d'une représentation synthétique par un mélange de gaussiennes. Nous utilisons donc maintenant un modèle de mélange comme abstraction de chaque image. Dans cette section, nous proposons un algorithme K-means pour réaliser la classification de la collection d'images.

L'algorithme K-means n'était originellement adapté qu'à l'optimisation d'une distorsion quadratique [McQueen, 1967, Lloyd, 1982]. Il a toutefois été généralisé plus récemment aux divergences de Bregman [Banerjee et al., 2005b,a], dont la divergence KL est une instance. Il est donc en principe possible d'appliquer l'algorithme K-means sur nos représentations.

Toutefois, cet algorithme nécessite la calcul de centroides selon une divergence de Bregman à chaque itération. Dans le cas de distributions de la famille exponentielle, Nielsen et al. ont démontré l'unicité de ces centroides (les mesures de divergence n'étant en général pas symmétriques, tous les cas possibles sont étudiés) [Nielsen and Nock, 2009, Nielsen and Boltz, 2010]. Toutefois, nous avons vu précédemment que la famille des mélanges de distributions ne fait en général pas partie de la famille exponentielle (voir section 3.3). Ainsi, il n'est pas possible de garantir l'unicité d'un centroide sur cette famille, compliquant ainsi singulièrement l'algorithme.

Nous effectuons une conjecture sur le calcul de ce centroide :

Conjecture 6.3.1 *Soit $\theta' = \{\theta'_1, \ldots, \theta'_L\}$ une collection de mélanges de gaussiennes.*

Alors un centroide approché (ou pseudo-centroide) g de θ' est obtenu en sortie de l'algorithme VBGMM-A, selon la divergence à gauche (i.e. $KL(g||\theta')$).

À l'appui de cette conjecture, remarquons que VBGMM-A trouve une approximation q à l'a posteriori réel p optimisant le critère $\mathrm{KL}(q||p)$ [Smidl and Quinn, 2006]. Partant de ce que p est l'a posteriori réel à l'origine de l'échantillon $\mathbf{X}$ traité par l'algorithme, et que, dans le cas de VBGMM-A, cet échantillon $\mathbf{X}$ est virtuel, et généré par θ', il est assez naturel de conjecturer que le modèle g estimé en sortie de VBGMM-A minimise $\mathrm{KL}(g||\theta')$, ou de manière équivalente que g est le centroide de θ'.

Algorithme K-means avec pseudo-centroides

Nous avons vu qu'un centroide de mélanges de gaussiennes n'était en général pas unique. De plus, VBGMM-A est sujet au minima locaux, et n'estime qu'une approximation. Aussi, la convergence de K-means n'est alors pas garantie. Nous verrons cependant, qu'empiriquement, le comportement de celui-ci est satisfaisant.

Le critère à optimiser est alors la somme des divergences KL à gauche des représentants d'images par rapport à leurs centroides respectifs (cette somme sera par la suite nommé distorsion). Nous proposons l'algorithme itératif suivant pour trouver un minimum local à ce coût :

Entrées : N images $\{f_n\}$, $n \in 1 \dots N$, nombre de centroides k
Sorties : Partitionnement des images en au plus k groupes
1 **Initialisation** : choisir k centroides $\{g_i\}$, $i \in 1 \dots k$ parmi les éléments ;
2 **tant que** *Affectations évoluent* **faire**
3 **Affecter** les images à une classe :
4 classe$(f_n) = \arg\min_i \mathrm{KL}(g_i \parallel f_n)$;
5 **Mettre** *à* **jour** les centroides :
6 regrouper les images selon leur valeur de $i = \text{classe}(f_n)$;
7 calculer les pseudo-centroides g_i en utilisant VBGMM-A ;
8 **fin**

Algorithme 7 : Algorithme K-means pour un minimum local de la distorsion

Affectation des images aux centroides

Dans la section précédente nous avons vu comment représenter chaque image par un mélange de gaussiennes, et comment obtenir un représentant depuis un groupe de mélanges de gaussiennes. Nous avons ainsi une implémentation pour la mise à jour des centroides.

Par ailleurs, les affectations d'images à une classe sont obtenues en minimisant la divergence KL entre les représentants d'images et de classes. Nous apportons maintenant quelques détails sur ce calcul. Il n'existe pas d'expression analytique pour la divergence entre de telles distributions, nous recherchons donc un compromis entre un calcul précis et un coût de calcul réduit. La solution classique consiste à estimer cette divergence par Monte-Carlo, mais le coût de calcul associé est trop élevé pour être envisagé.

Une approximation brute, proposée dans [Goldberger and Roweis, 2004], revient à associer chaque composante d'un des 2 mélanges à celle qui en est la plus proche dans

l'autre mélange, et d'approcher la divergence réelle par la somme de ces divergences entre paires de gaussiennes, quantités qui peuvent être facilement calculées. Une possibilité plus précise, que nous retenons dans le cas présent, est basée sur l'*unscented transform* [Goldberger et al., 2003], identifiée par $\mathrm{KL}_{\mathrm{UT}}$ dans la suite du document. Par contraste, nous noterons l'approximation Monte Carlo $\mathrm{KL}_{\mathrm{MC}}$.

Traditionnellement, l'approximation Monte-Carlo (i.e. par une somme finie) de la divergence KL est définie comme suit :

$$\mathrm{KL}_{\mathrm{MC}}(q||p) = \sum_{\mathbf{X}} \ln \frac{q(\mathbf{x}_i)}{p(\mathbf{x}_i)} \approx \int q(\mathbf{x}) \ln \frac{q(\mathbf{x})}{p(\mathbf{x})} d\mathbf{x} \tag{6.1}$$

avec $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ un échantillon généré depuis $q(\mathbf{x})$. L'idée fondatrice d'*unscented transform* est de choisir judicieusement un échantillon de taille faible.

Considérons d'abord le cas où $q(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu, \Sigma)$. Nous réalisons la diagonalisation de Σ ; nous obtenons alors $\Sigma = \mathbf{U}\mathbf{D}\mathbf{U}^T$, avec les vecteurs propres $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_d)$, et la matrice diagonale des valeurs propres $\mathbf{D} = diag(\lambda_1, \ldots, \lambda_d)$.

À cette distribution, nous associons un échantillon de $2d + 1$ points comme suit :

$$\mathbf{x}_i = \mu + \sqrt{\lambda_i}\mathbf{u}_i \; i = 1, \ldots, d \tag{6.2}$$
$$\mathbf{x}_{d+i} = \mu - \sqrt{\lambda_i}\mathbf{u}_i \; i = 1, \ldots, d \tag{6.3}$$
$$\mathbf{x}_{2d+1} = \mu \tag{6.4}$$

Nous utilisons cet échantillon dans la formule 6.1. Maintenant nous pouvons généraliser au cas où $q(\mathbf{x})$ est un mélange de gaussiennes, en impliquant les poids du mélange dans l'expression :

$$\mathrm{KL}_{\mathrm{UT}}(q||p) \approx \frac{1}{2d+1} \sum_{k=1}^{K} \omega_k \sum_{i=1}^{2d+1} \ln \frac{q(\mathbf{x}_{i,k})}{p(\mathbf{x}_{i,k})} \tag{6.5}$$

avec $\{\mathbf{x}_{i,k}\}$ l'ensemble de points obtenus à partir de la $k^{\text{ème}}$ composante de q.

Les biais des différentes méthodes d'estimation de divergences KL entre mélanges de gaussiennes ont été étudiés dans [Hershey and Olsen, 2007]. $\mathrm{KL}_{\mathrm{UT}}$ et $\mathrm{KL}_{\mathrm{MC}}$ figurent

parmi les approximations les plus satisfaisantes. Nous préfèrerons $\mathrm{KL}_{\mathrm{UT}}$ pour les raisons suivantes :

- cette approximation est bornée, ce qui est préférable du point de vue de l'implémentation,
- sa valeur est déterministe, contrairement $\mathrm{KL}_{\mathrm{MC}}$. Encore une fois, d'un point de vue algorithmique, cette caractéristique est préférable.

Notre système de visualisation est basé sur les similarités des centroides entre eux, traditionnellement comprises dans l'intervalle $[0, 1]$: aussi, la mesure de distorsion que nous venons de présenter serait difficilement utilisable. Afin de se conformer à l'intervalle requis pour les similarités, nous proposons d'utiliser la mesure suivante :

$$\mathrm{Sim}(g_i, g_j) = 1 - \mathrm{JS}(g_i, g_j) \tag{6.6}$$

avec $\mathrm{JS}(g_i, g_j)$ la divergence de Jensen-Shannon entre g_i et g_j, définie comme $\frac{1}{2}\big(\mathrm{KL}(g_i, m) + \mathrm{KL}(g_j, m)\big)$, où $m = \frac{1}{2}(g_i + g_j)$. Cette divergence peut être vue comme une version normalisée (i.e. $\mathrm{JS}(g_i, g_j) \in [0, 1]$) et symétrique (i.e. $\mathrm{JS}(g_i, g_j) = \mathrm{JS}(g_j, g_i)$) de la divergence KL.

L'algorithme K-means, appliqué rigoureusement, garantit une diminution monotone de la distorsion. Ayant eu recours à des approximations, nous risquons exceptionnellement d'observer de légères augmentations de notre distorsion, en lieu et place de légères diminutions. Ceci explique que nous ayons choisi d'observer les affectations (i.e. de mesurer l'agitation, voir formule (3.68)) pour évaluer la convergence.

6.3.3 Traitement incrémental

Considérons maintenant un flux de données entrantes. Dans ce contexte il est impossible de stocker et traiter un échantillon complet, nous devons donc définir une stratégie permettant de renouveler celui-ci. Des adaptations incrémentales de K-means ont été proposées [Bottou and Bengio, 1995], mais celles-ci reposent sur l'utilisation d'individus de type vecteur dans $\mathbb{R}^d$.

Alternativement, nous choisissons donc d'utiliser une fenêtre glissante, une solution classique dans ce genre de situation [Babcock et al., 2003]. Pour ce faire, nous définissons une taille de fenêtre N, qui contiendra les N éléments les plus récents. Les éléments dépassant ce cadre sont oubliés.

Le calcul de nos pseudo-centroides n'est pas basé sur une moyenne dans un espace Euclidien (voir section 6.3.1), aussi à chaque fois qu'un nouvel élément arrive dans le flux, nous devons mettre à jour les centroides et les affectations. Nous devrions alors

répéter l'opération jusqu'à convergence afin d'obtenir un minimum local.

Nous voyons toutefois que cette approche serait beaucoup trop coûteuse en termes de calcul. Nous pouvons alléger ce coût moyennant quelques simplifications et approximations :

- Au lieu de précéder à la mise à jour à chaque arrivée d'élément, nous pourrions l'effectuer tous les h éléments (en s'assurant tout de même que $h << N$, afin de garder une certaine stabilité).
- Au lieu d'aller jusqu'à convergence, et compte tenu qu'après un historique d'exécution significatif, l'agitation au sein des affectations est susceptible d'être assez faible, on pourrait arrêter les itérations une fois que la variation de distorsion diminue (i.e. le signe de la dérivée seconde diminue).

Ci-dessous, nous donnons l'adaptation algorithmique prenant en compte la deuxième simplification évoquée (figure 8).

Entrées : Nombre de centroides k
Sorties : Partition des données de la fenêtre glissante
1 **Initialisation** : k centroides = k premiers éléments arrivés ;
2 **tant que** *données entrantes* **faire**
3 | Ajouter éléments entrants au début de la fenêtre glissante ;
4 | **si** *fenêtre pleine* **alors**
5 | | enlever les éléments les plus anciens
6 | **fin**
7 | Itérer 7 jusqu'à $\frac{d(\Delta\text{distorsion})}{dt} > 0$;
8 | Afficher partition des éléments courants ;
9 **fin**

Algorithme 8 : Adaptation incrémentale de l'algorithme 7

6.4 Visualisation interactive des groupes

Notre méthode peut être utilisée en mode incrémental, permettant ainsi de réaliser le partitionnement de grandes bases de données. Mais dans ce contexte, il peut être difficile d'interpréter le flux de résultats produits par l'algorithme, ainsi que d'interagir avec. Nous proposons une méthode de visualisation résumant l'activité de partitionnement, et permettant à l'utilisateur d'agir sur celle-ci.

Afin de résumer l'information, nous ne visualiserons que les centroides, et les quelques images parmi les plus et les moins représentatives : ceci permet en effet à l'utilisateur de détecter les images éventuellement inadaptées à un groupe donné. Nous calculons

les similarités relatives entre centroides : ceci nous sert à construire un graphe complet dont les noeuds sont nos centres de groupes, et le poids de chaque arc est inversement proportionnel à la similarité entre les deux centroides reliés par cet arc.

Au cours du processus (en particulier incrémental), la partition et les centroides évoluent, nous avons donc besoin d'une représentation graphique qui transcrit cette évolution de manière expressive et agréable pour un utilisateur. Plusieurs méthodes ont été proposées dans la littérature pour la visualisation de séries temporelles.

Certains auteurs proposent une visualisation qui évolue à chaque itération comme un *morphing* (ou transformation lissée) entre l'ancienne représentation et la nouvelle représentation [Bender-deMoll and McFarland, 2006, Loubier et al., 2007, Kang et al., 2007]. Mais dans ce cas, le processus de visualisation suit plusieurs étapes : 1) fournir une représentation graphique et 2) réaliser le *morphing* de cette représentation avec la précédente. Toutefois, une série discrète de partitions peut présenter des discontinuités, ce qui rend cette technique de représentation difficilement utilisable.

Tenant compte de ces considérations, nous avons choisi d'utiliser un algorithme force et ressort [Kamada and Kawai, 1989, Fruchterman and Reingold, 1991] pour afficher notre graphe. Cette classe d'algorithmes (basée sur un positionnement influencé par l'application de forces) est souvent efficacement utilisée pour le dessin de graphes. L'approximation physique sous-jacente est facilement interprétée par un être humain. On a ainsi une méthode simple de visualisation de graphes, dont les noeuds sont en général bien distribués. Les symétries éventuelles du graphe sont conservées grâce à cette méthode de représentation.

Celui-ci est adapté à notre contexte de partitionnement, étant donné qu'il produit des positionnements des noeuds dans l'espace présentant une bonne qualité (éloignement des noeuds non-reliés, et rapprochement des noeuds reliés entre eux) pour des graphes de taille moyenne (jusqu'à 50-100 noeuds).

Parmi les algorithmes force et ressort, deux variantes distinctes existent :

- La première associe un ressort (ou plus précisément la force appliquée à celui-ci) à chaque paire de noeuds (i, j), la longueur idéale $d(i, j)$ de celui-ci étant alors inversement proportionnelle à la similarité effective entre les prototypes i et j. On résout alors un problème d'optimisation en minimisant la différence entre les longueurs idéales des ressorts, et les longueurs contraintes par la structure du graphe.
- La seconde associe toujours un ressort à chaque arc, mais de surcroit une charge électrique à chaque noeud. Le graphe complet est alors simulé en laissant évoluer ce système physique. A chaque itération nous calculons la somme des forces dans notre système en utilisant la loi de Hooke pour les arcs, et la loi de Coulomb

pour les noeuds. La première force cherche à maintenir chaque arc à une longueur appropriée, alors que la loi de Coulomb cherche à maintenir tous les noeuds loin les uns des autres.

Nous utilisons la deuxième variante pour notre système. En tant qu'évolution simulée d'un système physique, cette méthode nous permet de produire un affichage continu, et parlant pour le système perceptuel humain.

6.4.1 Algorithme force et ressort, et amélioration de la visualisation

Afin de permettre à l'utilisateur de visualiser la construction continue des groupes, nous proposons un algorithme force et ressort incrémental pour la mise à jour de notre graphe de centroides en 3D. Ce type d'algorithmes a déjà été utilisé pour la visualisation interactive de graphes [McCrickard and Kehoe, 1997]. L'utilisateur peut alors naturellement comprendre l'évolution du tracé du graphe. Cette évolution lui est présentée étape par étape, proposant ainsi un moyen d'interaction. Assez naturellement, l'utilisateur peut modifier manuellement la position de noeuds dans l'espace, impliquant ainsi une modification du système à optimiser.

À chaque itération de l'algorithme de visualisation, la répulsion de Coulomb (voir équation 6.7, avec q_1 et q_2 les charges électriques de 2 noeuds n_1 and n_2, $\vec{r}$ le vecteur entre n_1 et n_2), et l'attraction de Hooke (voir équation 6.8, avec $\vec{L}$ un vecteur entre les deux noeuds reliés par l'arc, et R la longueur idéale du ressort) du réseau sont calculées, et la position de chaque noeud est mise à jour (voir algorithme 9).

$$
\begin{aligned}
E_0 &= 8.85 \cdot 10^{-12} * C^2/(N * m^2) \\
\vec{F_{\text{Coulomb}}} &= \frac{q_1 \cdot q_2 \cdot \vec{r}}{4\pi E_0 \left|\vec{r}\right|^3} && (6.7) \\
\vec{F_{\text{Hooke}}} &= -kHooke\left(|\vec{L}| - R\right)\frac{\vec{L}}{|\vec{L}|} && (6.8)
\end{aligned}
$$

Dans cet algorithme, $V_{n_i}^t$ représente la vitesse du noeud n_i à l'instant t ; $P_{n_i}^t$ représente la position du même noeud au même instant ; TempsEtape représente le temps écoulé entre 2 itérations de l'algorithme ; et ParamètreEchelle est une constante fixée à 0.2 expérimentalement.

Ces formules nous permettent de mettre à jour les caractéristiques du graphe en parallèle, et ainsi de prendre en compte les changements de partition produits par l'algorithme dédié. De cette manière, nous pouvons présenter une animation lissée à l'utilisateur. Enfin, nous affichons des miniatures du sous-ensemble d'images les plus proches de chaque centroide sur chaque noeud associé.

```
pour tous les Noeud n_i ∈ ListeDeNoeuds faire
    forceRéseau ← 0;
    pour tous les Noeud n_j ∈ ListeDeNoeuds\n_i faire
        forceRéseau ← forceRéseau + RépulsionDeCoulomb(n_i, n_j) ;
    fin
    ;
    pour tous les Noeud n_j ∈ AarcVers(n_i) faire
        forceRéseau ← forceRéseau + AttractionDeHooke(n_i, n_j) ;
    fin
    ;
    V^t_{n_i} ← (V^{t-1}_{n_i} + TempsEtape × forceRéseau) × ParamètreEchelle ;
    P^t_{n_i} ← P^{t-1}_{n_i} + V^t_{n_i} × TempsEtape ;
fin
```

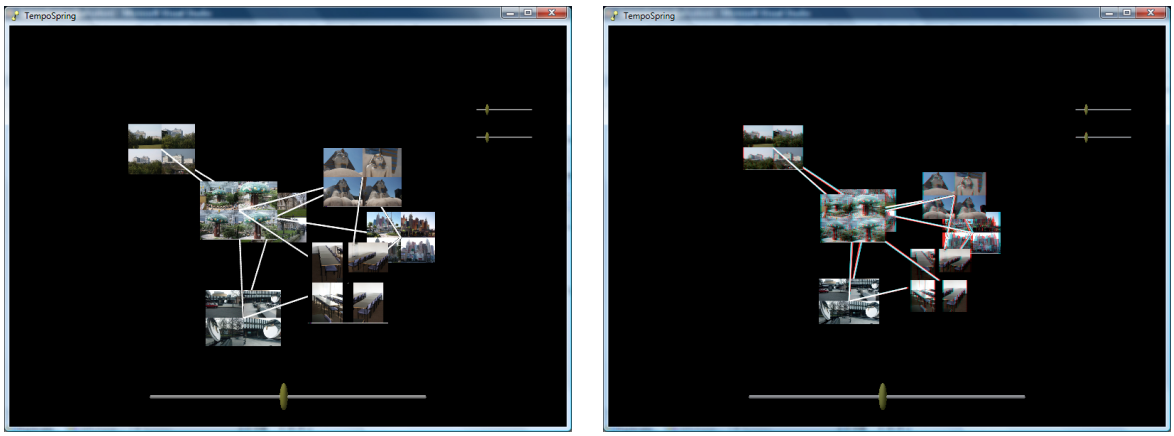
Algorithme 9 : Algorithme force-ressort incrémental

La figure 6.1 montre un exemple de disposition de notre visualisation. Nous remarquons que les miniatures représentation les centroides sont toujours affichées face à la caméra utilisateur (principe du *billboard* 3D).

Plusieurs améliorations ont été effectuées sur notre processus de visualisation. Afin de pouvoir observer la totalité du graphe à l'écran, à chaque mise à jour du graphe le centre de gravité de celui-ci est utilisé pour le recentrer à l'écran. Mais quand ce centre de gravité est amené à changer brutalement (e.g. quand le nombre de groupes change), l'opération de recentrage provoque une discontinuité de l'affichage. Pour éviter cela, le recentrage est effectué progressivement. L'accélération de cette opération est limitée, est la vitesse du déplacement est diminuée de manière constante à mesure que le centre de gravité et le centre de l'écran coïncident.

Afficher un graphe complet peut être difficile, et réduire la lisibilité générale de notre visualisation. D'autre part, afficher un lien entre deux groupes très dissimilaires n'apporte pas d'information. Mais le seuil qui indiquerait si un arc doit être tracé ou non est délicat à déterminer a priori. Nous proposons plutôt à l'utilisateur la possibilité de régler ce paramètre au moyen d'un potentiomètre (ou *slider*, voir fig. 6.1 en bas de la fenêtre). La limite basse à ce seuil est la valeur la plus petite pour laquelle le graphe reste connecté (condition nécessaire à la mise en oeuvre de l'algorithme force et ressort).

Finalement, des recouvrements peuvent survenir en 3D, ce qui peut rendre difficile la perception de la profondeur de notre visualisation. Pour contourner ce problème, nous avons implémenté notre application de manière à pouvoir être exécutée dans un environnement stéréoscopique, plusieurs technologies étant possibles (sous forme d'anaglyphes, ou de stéréoscopie avec lunettes polarisées).



a) *b)*

FIG. 6.1 – Capture d'écran de notre application en mode *a)* normal et *b)* anaglyphe. Les deux *sliders* à droite de l'écran permettent de régler respectivement la force de Coulomb et la force de Hooke. Le *slider* en bas de l'écran permet de filtrer les arcs de notre graphe

6.4.2 Interaction et rétro-action sur l'algorithme de partitionnement

Le résultat final de l'algorithme force et ressort peut être fortement influencé par la disposition initiale. Dans certains cas (et en particulier quand les arcs sont peu filtrés), cela peut conduire à une visualisation de faible qualité. Toutefois, dans notre application, l'utilisateur peut imprimer une rotation au graphe, ou sélectionner un groupe afin d'obtenir des détails spécifiques, comme par exemple le nombre d'éléments de celui-ci. Une fois sélectionné, l'utilisateur peut le déplacer dans l'espace 3D pour voir comment il interagit avec d'autres groupes, ou pour servir de point de départ d'une réorganisation de graphe. Pour pouvoir déplacer des éléments dans un espace 3D avec une simple souris, ceux-ci sont limités aux plans $\{x, y\}$, $\{x, z\}$ ou $\{y, z\}$ (voir figure 6.2).

Pour limiter le nombre d'objets à l'écran, nous n'affichons sur chaque noeud que des miniatures des 4 images les plus représentatives du groupe. En sélectionnant un groupe spécifique, l'utilisateur peut accéder à toutes les images associées (voir fig. 6.3).

Si l'algorithme produit une partition inconsistante avec les souhaits de l'utilisateur (ou une vérité terrain), celui-ci a trois moyens de rétroaction :

- en glissant une image hors du groupe g_i. À l'itération suivante, l'algorithme de partitionnement calculera un nouveau centroide pour g_i sans cet élément. Un nouveau groupe est créé avec l'élément glissé comme centre,
- en glissant une image d'un groupe g_i vers un nouveau groupe g_j. A l'itération suivante, l'algorithme de partitionnement met à jour les centroides concernés avec ces changements,

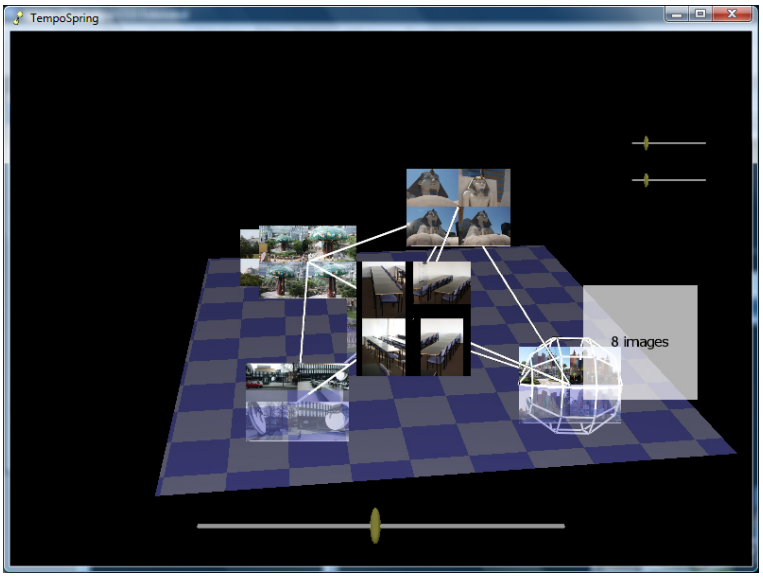


FIG. 6.2 – Illustration de l'interactivité de l'application de visualisation.

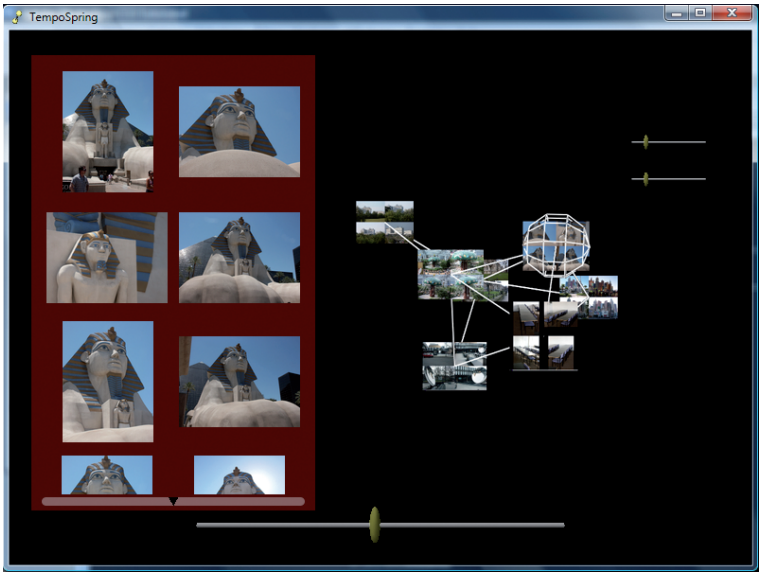


FIG. 6.3 – Rétroaction interactive sur l'algorithme de classification.

– en supprimant un groupe g_i. A l'itération suivante, les éléments précédemment affectés à celui-ci sont associés au groupe le plus proche parmi ceux restants.

6.5 Expériences

Dans cette section, nous allons nous concentrer sur l'évaluation de la version standard de notre algorithme de partitionnement (voir paragraphe 6.3.2). L'objectif est alors d'évaluer le comportement de notre méthode sur un ensemble d'images réelles, indépendamment des effets qui pourraient être induits par les variantes incrémentale et interactive. Pour nos expériences, nous avons sélectionné un échantillon de la collection d'images UCID [Schaefer and Stich, 2004].

Cette collection contient des photos réelles (bâtiments, personnes et objets, dans des conditions de prise de vue usuelles). Toutes ces images ont les mêmes dimensions. Chaque photo est associée avec une *vérité terrain*, i.e. un thème ou une personne spécifique, définissant ainsi des classes. Pour nos tests, nous avons sélectionné 134 images de cette collection, associées à 11 classes disjointes. Afin de conserver une complexité calculatoire acceptable, nous avons sous-échantillonné nos images (128x96 pixels).

Nous allons évaluer la capacité de notre algorithme à retrouver les 11 classes réelles, ou, tout au moins, d'autres groupes pertinents. Nous initialisons l'algorithme avec 11 représentations d'images choisies au hasard comme centroides. Les résultats présentés sont obtenus en tant que moyenne de 20 exécutions du même protocole. Sur la figure 6.4, nous avons surveillé l'évolution de la distorsion totale (voir section 6.3.2). Comme référence, nous avons ajouté une ligne pointillée indiquant la distorsion observée au sein des groupes réels.

Sur la figure 6.5, nous avons indiqué l'évolution du nombre de groupes estimés au cours de l'exécution du processus. En effet, même si l'algorithme est initialisé avec 11 groupes, la géométrie non-euclidienne des éléments traités rend possible un centroide sans image associée. Dans ce cas, le groupe de ce centroide disparait.

Sur la figure 6.6, nous avons relevé la dispersion des centroides. Afin d'être homogène avec la distorsion mesurée, nous avons utilisé une version symétrisée de la divergence KL, $\frac{1}{2}(\mathrm{KL}(p||q) + \mathrm{KL}(q||p))$, mais la mesure de similarité présentée en fin de section précédente aurait pu également être employée. Finalement, nous avons mesuré l'erreur

couple [Fowlkes and Mallows, 1983, Azzag, 2005, Picarougne et al., 2007] de notre partition relativement à la vérité terrain. Cette mesure d'erreur augmente seulement si deux éléments appartenant au même groupe réel sont affectés à deux groupes différents par notre algorithme, et réciproquement.

De ces résultats, nous formons les conclusions suivantes :

- Sur 20 expériences, nous obtenons une erreur couple moyenne de 11%.
- Comme attendu, la figure 6.4 suggère que la distorsion totale décroit très rapidement, et après 5 itérations converge lentement vers une valeur limite. Dans le cadre expérimental utilisé (échantillon statique, initialisations aléatoires), le niveau de distorsion observé dans les vrais groupes a presque pu être atteint. Avec l'intervention de l'interaction utilisateur, le processus d'estimation devrait donc être capable de fortement se rapprocher des vrais groupes.
- Des groupes disparaissent parfois au cours du processus (figure 6.5). Mais le nombre de groupes actifs reste toujours relativement proche de celui initialement suggéré, il n'y a pas de réduction drastique du nombre de groupes.
- La dispersion des centroides croit très rapidement au cours de l'exécution (figure 6.6). Autrement dit, nos groupes sont de mieux en mieux séparés. Nous remarquons que les centroides suggérés par les groupes réels sont moins bien séparés, en moyenne, que ceux que nous estimons avec notre algorithme. La prise en compte d'interactions devrait pouvoir permettre de dépasser ces minima locaux, grâce aux possibles recombinaisons proposées par l'utilisateur, pour éventuellement atteindre de meilleures solutions (comme suggéré en pointillé sur la figure 6.4).

6.6 Conclusion

Dans ce chapitre, nous avons présenté un système de classification d'une collection d'images, dans l'optique d'un outil de navigation pour un utilisateur. Dans un premier temps, nous avons proposé de représenter chaque image par un modèle probabiliste, et décrit un algorithme K-means pour partitionner de tels objets, avec une adaptation au cas incrémental.

Les classes obtenues et leurs similarités sont utilisées pour former un graphe évoluant dans le temps, que nous combinons à une technique de visualisation 3D adaptée. Nous avons envisagé l'interaction de l'utilisateur, et notamment la rétro-action sur le processus de partitionnement. Les actions de l'utilisateur seraient facilement incorporées dans l'algorithme, car celui-ci réalise une optimisation locale itérative ; une action change simplement la «zone» optimisée. Toutefois, ces aspect n'ont pas été pleinement développés ; nos résultats expérimentaux mesurent donc simplement la qualité de la méthode de classification proposée. Notons qu'au-delà de l'influence de l'utilisateur, le graphe change également pour refléter l'évolution itérative de l'algorithme de classification.

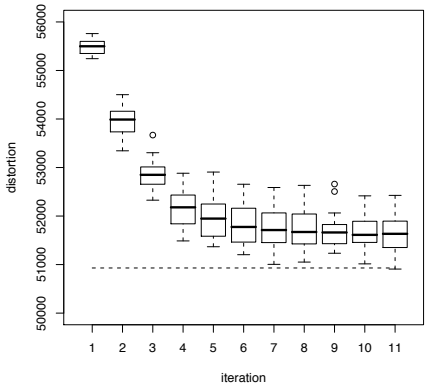


FIG. 6.4 – Evolution de la distorsion totale au cours des 10 premières itérations (moyenne sur 20 expériences). La ligne pointillée indique la distorsion parmi les groupes réels (représentation de Box et Whisker : la ligne en gras est la médiane, les autres lignes les principaux quantiles, les valeurs extrêmes sont indiquées par des cercles).

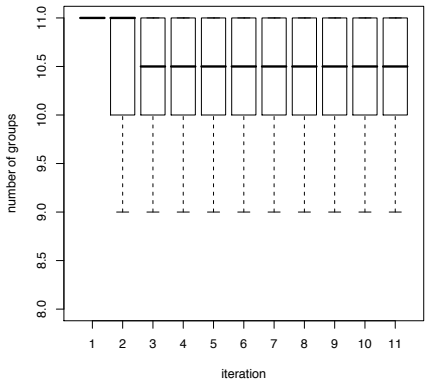


FIG. 6.5 – Evolution du nombre de groupes au cours des 10 premières itérations (20 exécutions).

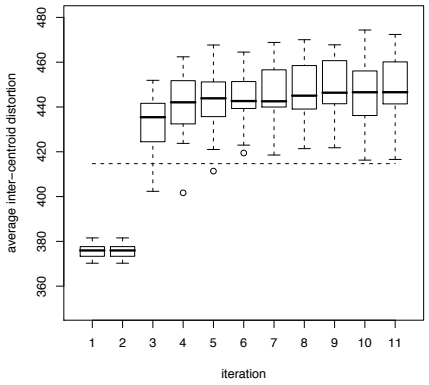


FIG. 6.6 – Evolution des métriques entre centroides moyennes au cours des 10 premières itérations (20 exécutions). La ligne pointillée indique cette mesure pour les groupes réels.

Chapitre 7

Conclusion générale

7.1 Bilan

Nous avons commencé par proposer un état de l'art des questions d'apprentissage de modèles de classification, afin de pouvoir situer nos travaux ultérieurement. Ensuite, nous avons présenté de manière extensive l'approche génératrice à l'apprentissage de modèles de mélange. Ce développement est en effet nécessaire pour envisager clairement l'apport de l'approche variationnelle dans ce contexte.

Cette approche bayésienne a servi de cadre théorique pour nos contributions sur l'agrégation de modèles de mélanges. Nous avons situé l'intérêt de cette tâche dans le cadre, plus vaste, de l'apprentissage sur des données distribuées. Les propriétés de l'approche variationnelle nous ont permis de mettre l'accent sur la parcimonie des solutions trouvées par nos algorithmes, un aspect important pour le passage à l'échelle. Nous avons proposé deux manières complémentaires de réduire la complexité des problèmes trouvés : en limitant le nombre de composantes en sortie (algorithme VBGMM-A, voir section 4.3), et en limitant la dimensionalité du sous-espace spécifique à chaque composante (algorithme VBMPPCA-A, voir section 4.5).

De même, nous avons proposé une vision probabiliste bayésienne d'un apprentissage semi-supervisé, afin de permettre la prise en compte de contraintes dans le processus d'agrégation.

Enfin, nous avons proposé deux approches visuelles et interactives originales à la classification d'une collection de données :

- Avec un algorithme bio-mimétique, privilégiant les interactions d'agents dans un espace 2D. On propose ainsi une interface intuitive, incluant l'utilisateur dans le processus.
- Avec un système de visualisation de classification, combinant nos techniques d'agrégation de modèles de mélange, une méthode de classification incrémentale, et une visualisation reproduisant la dynamique d'un système physique.

7.2 Perspectives

7.2.1 Utilisation de descripteurs d'images

L'estimation de modèles de mélanges sur les couleurs de pixels d'une image (réalisée, par exemple, dans la section 6.3.1) permet d'obtenir une bonne expressivité, mais limite l'usage aux images ayant peu de bruit de fond (e.g. pas d'images d'un objet spécifique à des échelles et décors de fond variés).

L'utilisation de descripteurs d'images (e.g. SIFT [Lowe, 1999], GIST [Torralba et al., 2008]) permet de se rapprocher d'un traitement sémantique de l'image, en détectant notamment des motifs indépendamment de l'échelle et de l'angle de ceux-ci. L'estimation de modèles de mélanges sur l'ensemble de descripteurs pouvant être extrait d'une image permettrait ainsi d'obtenir une représentation synthétique plus proche du contenu sémantique de celle-ci. Ces descripteurs ont une dimensionalité très élevée, aussi une piste complémentaire serait d'appliquer un mélange de MPPCA sur de tels ensembles de descripteurs. Nous aurions ainsi une méthode d'extraction de «mots visuels».

Cette piste est activement suivie par la communauté de recherche [Wu et al., 2010, Hotta, 2010, Inoue et al., 2010]. Ces mots visuels sont typiquement exploités dans le cadre d'un modèle de type LDA (voir section 3.5.5), permettant ainsi d'effectuer de la classification de collections d'images de manière bien fondée. Par ailleurs, à l'instar des modèles de mélange étudiés dans ce document, il est envisageable de compléter cette approche avec des contraintes semi-supervisées du type de celles présentées dans la section 4.7.

7.2.2 Extension du principe variationnel à d'autres modèles

Nous n'avons pas décliné les versions variationnelles incrémentales (ou de filtrage bayésien, voir section 3.5.5) des algorithmes VBGMM et VBMPPCA. De même, des adaptations incrémentales de nos méthodes d'agrégation bayésienne n'ont pas été proposées dans ce document. Toutefois, l'application directe des principes spécifiés dans la section 3.5.5 devrait permettre d'obtenir ces algorithmes. D'autre part, les principes utilisés pour l'agrégation dans ce document pourraient a priori être adaptés sans problème à des modèles de mélange discrets tels que LDA, à condition que des distributions de probabilité de la famille exponentielle y soient utilisées.

7.2.3 Optimisations

L'algorithme du chapitre 6 met en jeu de nombreux calculs de divergence KL entre mélanges de gaussiennes. Nous avons vu que cette mesure n'a pas d'expression analytique, et peut seulement être estimée au moyen d'approximations. Cependant, les deux approximations présentées dans ce document, par Monte-Carlo ou *unscented transform*,

sont très coûteuses à calculer. Il serait intéressant d'étudier les optimisations possibles pour la réalisation de ces mesures (e.g. utiliser des éléments pré-calculés pour l'*unscented transform*).

Dans le même chapitre, la construction des centroides de mélanges de gaussiennes à chaque itération est également assez couteux. Cependant, en employant une variante incrémentale de notre algorithme de fusion de mélanges (évoquée dans la section 7.2.2, cette étape devrait être beaucoup plus rapide.

Les MPPCA sont parcimonieux en paramètres, mais leur estimation implique un important surplus de la taille des variables latentes impliquées. De même, ce choix de modélisation est à l'origine indirecte de la difficulté du paramétrage de l'algorithme variationnel associé. Mécaniquement, l'agrégation souffre du même problème, même si il est limité par notre connaissance a priori de la structure des données traitées.

Nous avons proposé une solution simple, consistant à limiter le rang maximal des composantes à estimer. D'autres solutions ont été esquissées dans la littérature :

- Un modèle simplifié a été proposé dans [Smidl and Quinn, 2006]. Cependant, cette proposition ne comporte qu'une composante ACP, il faudrait donc au préalable étudier les possibilités d'extension au MPPCA,
- En utilisant le formalisme proposé dans [Watanabe et al., 2009]. Toutefois, les composantes ACP seraient alors contraintes de partager le même sous-espace.

7.2.4 Intégration de contraintes

Nous avons proposé un modèle pour intégrer des contraintes au processus d'agrégation de modèles de mélange. Sans perte de généralité, il est possible d'adapter ce modèle à l'algorithme VBGMM, et d'autoriser des contraintes définies par un utilisateur. Le modèle menant à l'algorithme VBMPPCA-A pourrait également être étendu de manière analogue.

Par souci de simplicité, notre algorithme VBGMM-B fait une hypothèse de précédence assez forte dans l'étape E ; on pourrait relaxer quelque peu celle-ci en effectuant cette étape par échantillonnage via la méthode de Gibbs (*Gibbs sampling* [Geman and Geman, 1984]). L'utilisation d'étapes stochastiques dans le contexte d'un algorithme EM a en effet déjà été étudié avec succès dans la littérature [Celeux and Govaert, 1992]. Le coût calculatoire supplémentaire, et l'amélioration qualitative, sont cependant difficiles à évaluer a priori.

7.2.5 Amélioration de l'opérateur d'éclatement

Notre adaptation incrémentale de méthode bio-mimétique basée sur les mouvements d'agents présente des résultats convaincants, mais nous avons vu que la prise en compte

d'étiquettes fournies par un utilisateur n'apporte qu'une amélioration marginale.

Pour pallier cette situation, nous pourrions améliorer l'opérateur d'éclatement en utilisant une coupe non-symétrique de la gaussienne associée à un agent le long d'un des vecteurs propres de covariance. Il serait également souhaitable d'utiliser une mesure de similarité entre agents qui prenne en compte les matrices de covariance des distributions en jeu. Une mesure de similarité basée sur la divergence KL serait une bonne candidate dans cette optique. Il serait par ailleurs intéressant d'appliquer l'opérateur d'éclatement quand l'entropie d'une distribution atteint un certain seuil qui serait calculé dynamiquement.

7.2.6 Interactions et rétroactions

De manière générale, l'interaction entre un utilisateur, et les processus de classification présentés dans ce documents, aurait pu être approfondie. En particulier, dans le contexte de la visualisation de classification (section 6), la rétroaction consiste simplement à réarranger les groupes : on pourrait aussi en tirer de l'information (e.g. utiliser une métrique adaptive en fonction des préférences de similarité de l'utilisateur [Basu et al., 2004]). Nous pourrions également y associer des mécanismes de fusion et d'éclatement, afin d'exploiter des étiquettes qui seraient fournies dynamiquement par l'utilisateur.

7.2.7 L'agrégation dans le contexte d'un système de transmission ou d'analyse de sources

Nos techniques d'agrégation sont en quelque sorte focalisées sur la qualité de restitution d'un signal : pour en réaliser des applications pertinentes, il serait nécessaire de combiner nos mécanismes avec des approches pour la construction et la diffusion de modèles de qualité sur les noeuds d'un réseau [Nikseresht and Gelgon, 2008], ou d'analyse de sources de données (prenant par exemple en compte la redondance [Cabanes and Bennani, 2009], ou la fiabilité de celles-ci).

Bibliographie

K. Aihara and A. Takasu. Category based customization approach for information retrieval. *Lecture Notes in Computer Science*, 2109/2010 :207–209, 2010.

N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information : Ranking and clustering. *Journal of the ACM*, 55(5), 2008.

H. Akaike. A new look at the statistical model identification. *IEEE Trans. on Automatic Control*, AC-19(6), 1974.

F. Alimoglu. Combining multiple classifiers for pen-based handwritten digit recognition. Technical report, Institute of Graduate Studies in Science and Engineering, 1996.

A. Asuncion and D. Newman. Uci machine learning repository. *http ://archive.ics.uci.edu/ml/*, 2007.

H. Attias. A variational bayesian framework for graphical models. *Advances in Neural Information Processing Systems - MIT Press*, 12, 2000.

H. Azzag. *Classification hiérarchique par des fourmis artificielles : application à la fouille de données et de textes pour le Web.* PhD thesis, Ecole Doctorale Santé Sciences et Technologies, Université François Rabelais Tours, 2005.

H. Azzag, C. Guinot, and G. Venturini. Anttree : web document clustering using artificial ants. *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 04)*, pages 480–484, 2004.

H. Azzag, C. Guinot, and G. Venturini. *Swarm Intelligence and Data Mining*, volume 34 of *Studies in Computational Intelligence*, chapter 7 - Data and text mining with hierarchical clustering ants. Springer-Verlag, 2006a.

H. Azzag, D. Ratsimba, D. Da Costa, C. Guinot, and G. Venturini. On building maps of web pages with a cellular automata. *IFIP Conference on Biologically Inspired Cooperative Computing WCC-BICC*, 2006b.

B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining variance and k-medians over data stream windows. *Proceedings of the 22nd ACM SIGMOD-SIGACT symposium on Principles of database systems*, pages 234–243, 2003.

S. Baluja and M. Covell. Beyond "near-duplicates" : Learning hash codes for efficient similar-image retrieval. *Proceedings of IAPR/IEEE International Conference on Pattern Recognition (ICPR'10)*, 2010.

A. Banerjee, X. Guo, and H. Wang. On the optimality of conditional expectation as a bregman predictor. *IEEE Trans. on Information Theory*, 51(7) :2664–2669, 2005a.

A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6 :1705–1749, 2005b.

S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68, 2004.

S. Basu, M. Bilenko, A. Banerjee, and R. J. Mooney. Probabilistic semi-supervised clustering with constraints. In *Semi-Supervised Learning*. MIT Press, 2006.

M. J. Beal. *Variational Algorithms for approximate inference*. PhD thesis, University of London, 2003.

M. Bechchi, G. Raschia, and N. Mouaddib. Merging distributed database summaries. *Conference on Information and Knowledge Management*, pages 419–428, 2007.

Skye Bender-deMoll and Daniel A. McFarland. The art and science of dynamic network visualization. *Journal of Social Structure*, 7(2), 2006.

C. Biernacki, G. Celeux, and G. Govaert. Choosing starting values for the em algorithm for getting the highest likelihood in multivariate gaussian mixture model. *Computational Statistics and Data Analysis*, 2003.

C. M. Bishop. Variational principal components. *Proceedings of 9th ICANN*, 1 :509–514, 1999.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

C. M. Bishop and M. Svensen. The generative topographic mapping. *Neural Computation*, 10(1) :215–234, 1998.

Christopher M. Bishop and Markus Svensén. Bayesian hierarchical mixtures of experts. In *UAI*, pages 57–64, 2003.

R. E. Blahut. *Principles and Practice of Information Theory*. Addison-Wesley, 1987.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3 :993–1022, 2003.

K. Blekas and I. E. Lagaris. Split-merge incremental learning (smile) of mixture models. *ICANN'07*, Part II(LNCS 4669) :291–300, 2007.

E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence : From Natural to Artificial Systems*. Oxford University Press, New York, 1999.

L. Bottou and Y. Bengio. Convergence properties of the k-means algorithm. *Advances in Neural Information Processing Systems - MIT Press*, 7, 1995.

L. Breiman, J. H. Friedman, R. A. Olshen, and P. J. Stone. *Classification and Regression trees*. Wadsworth, 1984.

P. Bruneau. **De l'utilisation pratique des mélanges de gaussiennes**. *Journée des doctorants de l'école doctorale STIM (JDOC'09) - prix du meilleur article (sur env. 100)*, Nantes (France), Avril 2009.

P. Bruneau, M. Gelgon, and F. Picarougne. **Parameter-Based Reduction of Gaussian Mixture Models with a Variational-Bayes Approach**. *Actes de IEEE/IAPR International Conference on Pattern Recognition (ICPR'08)*, Tampa (USA), Décembre 2008a.

P. Bruneau, F. Picarougne, and M. Gelgon. **Incremental clustering in a data stream with a flock of agents**. *Actes de joint meeting of the Société Francophone de Classification and the Classification and Data Analysis Group of the Italian Society of Statistics (SFC-CLADAG'08)*, Naples (Italie), Juin 2008b.

P. Bruneau, A. Pigeau, M. Gelgon, and F. Picarougne. **Geo-temporal structuring of a personal image database with two-level variational-Bayes mixture estimation**. *LNCS 5811 - Revised selected papers from Adaptive Multimedia Retrieval Workshops (AMR'08-AMR'09)*, pages 127–139, 2008c.

P. Bruneau, M. Gelgon, and F. Picarougne. **Parsimonious variational-Bayes mixture aggregation with a Poisson prior**. *Actes de European Signal Processing Conference (EUSIPCO'2009)*, Glasgow (UK), Août 2009a.

P. Bruneau, F. Picarougne, and M. Gelgon. **Incremental semi-supervised clustering in a data stream with a flock of agents**. *Actes de IEEE Congress on Evolutionary Computing (CEC'2009)*, Trondheim (Norvège), Mai 2009b.

P. Bruneau, M. Gelgon, and F. Picarougne. **Parsimonious reduction of Gaussian mixture models with a variational-Bayes approach**. *Pattern Recognition*, 43 : 850–858, Mars 2010a.

P. Bruneau, M. Gelgon, and F. Picarougne. **A variational-Bayes technique for aggregating probabilistic PCA mixtures from their parameters**. Rapport de recherche, version étendue de ICPR'2010, 30 pages, soumis à une revue, http ://hal.archives-ouvertes.fr/docs/00/47/60/76/pdf/pami.pdf, HAL, Juin 2010b.

P. Bruneau, M. Gelgon, and F. Picarougne. **Aggregation of probabilistic PCA mixtures with a variational-Bayes technique over parameters**. *Actes de IEEE/IAPR International Conference on Pattern Recognition (ICPR'10)*, Istanbul (Turquie), Août 2010c.

P. Bruneau, M. Gelgon, and F. Picarougne. **Fusion bayésienne de mélanges de gaussiennes par une approche variationnelle**. *Actes de Reconnaissance de Formes et Intelligence Artificielle (RFIA'10)*, nommé parmi les 5 meilleurs articles, Caen (France), Janvier 2010d.

P. Bruneau, F. Picarougne, and M. Gelgon. **Interactive unsupervised classication and visualization for browsing an image collection**. *Pattern Recognition*, 43 (2) :485–493, 2010e.

P. Bruneau, F. Picarougne, and M. Gelgon. **Visualisation dynamique de classification d'images semi-supervisée incrémentale**. *Actes de joint meeting of the Société Francophone de Classification and the Classification and Data Analysis Group of the Italian Society of Statistics (SFC-CLADAG'10)*, La Réunion (France) 2010f.

H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern recognition Letters*, 19(3-4) :255–259, 1998.

C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2) :121–167, 1998.

G. Cabanes and Y. Bennani. A local density-based simultaneous two-level algorithm for topographic clustering. *IEEE International Joint Conference on Neural Networks*, pages 1176–1182, 2008.

G. Cabanes and Y. Bennani. Comparing large datasets structures through unsupervised learning. *LNCS 5863 - Neural Information Processing*, pages 546–553, 2009.

J. E. Camargo, J. C. Caicedo, A. M. Chavarro, and F. A. Gonzaléz. A kernel-based strategy for exploratory image collection search. *International Workshop on Content-Based Multimedia Indexing*, pages 1–6, 2010.

P. Carbonetto, G. Dorko, C. Schmid, H. Kück, and N. de Freitas. Learning to recognize objects with little supervision. *International Journal of Computer Vision*, 77(1-3) : 219–237, 2008.

C. Carson, S. Belongie, H. Greenspan, and J. Malik. Region-based image querying. *Proceedings of IEEE Workshop on Content-Based access of Image and Video libraries*, pages 42–49, 1997.

C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld : A system for region-based image indexing and retrieval. *Third international conference on Visual Information Systems*, 1999.

C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld : Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Learning*, 24(8) :1026–1038, 2002.

G. Celeux and G. Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, 1992.

G. Celeux, S. Chretien, F. Forbes, and A. Mkhadri. A component-wise EM algorithm for mixtures. *Journal of Computational and Graphical Statistics*, 10(4) :697–712, 2001.

O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

Y. Chen, J. Z. Wang, and R. Krovetz. Content-based image retrieval by clustering. *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, POSTER SESSION :193–200, 2003.

M. Clerc. *L'optimisation par essaims particulaires : versions paramétriques et adaptatives*. Hermes-Lavoisier, 2005.

W. Cleveland. *Visualizing Data*. Hobart Press, 1993.

S. E. Coull, F. Monrose, M. K. Reiter, and M. Bailey. The challenges of effectively anonymizing network data. *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*, pages 230–236, 2009.

R. Cucchiara. Analysis and comparison of different genetic models for the clus- tering problem in image analysis. *International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 423–427, 1993.

R. Datta, J. Li, and J. Z. Wang. Content-based image retrieval : approaches and trends of the new age. *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 253–262, 2005.

L. N. De Castro and F. J. Von Zuben. An evolutionary immune network for data clustering. *Proceedings of the IEEE SBRN'00 (Brazilian Symposium on Artificial Neural Networks)*, pages 84–89, 2000.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society - B Series*, B (39) :1–38, 1977.

J. L. Deneubourg, S. Goss, N. R. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting : robot-like and ant-like robots. *Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 356–365, 1990.

T. G. Dietterich. Ensemble methods in machine learning. *LNCS 1857 - Multiple Classifier Systems*, pages 1–15, 2000.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. *Proceedings of the 24th VLDB Conference*, 1998.

E. Falkenauer. A new representation and operators for genetic algorithms applied to grouping problems. *Evolutionary Computation*, 2(2) :123–144, 1994.

R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. *Advances in Neural Information Processing Systems - MIT Press*, 22 : 522–530, 2009.

D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2 :139–172, 1987.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(Part II) :179–188, 1936.

L. Fogel, A. Owens, and M. Walsh. *Artificial Intelligence Through Simulated Evolution*. Wiley, 1966.

E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *J. Am. Stat. Assoc.*, 78 :553–569, 1983.

Y. Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3) :293–318, 2001.

Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11) :1129–1164, 1991. ISSN 0038-0644.

V. Garcia and F. Nielsen. Simplification and hierarchical representations of mixtures of exponential families. *Signal Processing*, 90 :3197–3212, 2010.

V. Garcia, F. Nielsen, and R. Nock. Hierarchical gaussian mixture models. *Proceedings of IEEE Conference on Acoustics, Speech and Signal Processing*, 2010.

M. Gelgon, A. Maillet, P. Bruneau, F. Picarougne, and R. Lehn. **The Safim@ge Platform**. *Actes de Network and Electronic Media Forum (NEM'2008)*, Paris (France) 2008.

S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis Machine Intelligence*, 6(1) : 721–741, 1984.

Z. Ghahramani and M. J. Beal. Variational inference for bayesian mixtures of factor analysers. *Advances in Neural Information Processing Systems*, 2000.

G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints. *ACM Transactions on Database Systems (TODS)*, 34(2) :9, 2009.

A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

J. Goldberger and S. Roweis. Hierarchical clustering of a mixture model. *Advances in Neural Information Processing Systems - MIT Press*, 17, 2004.

J. Goldberger, S. Gordon, and H. Greenspan. An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures. *Proceedings of the 9th IEEE International Conference on Computer Vision*, 1 :487–493, 2003.

S. Goss and J. L. Deneubourg. Harvesting by a group of robots. *Proceedings of the First European Conference on Artificial Life*, pages 195–204, 1991.

H. Greenspan, J. Goldberger, and L. Ridel. A continuous probabilistic framework for image matching. *Computer Vision and Image Understanding*, 84(3) :384–406, 2001.

G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset, http ://authors.library.caltech.edu/7694. Technical Report 7694, California Institute of Technology, 2007.

GSL. GNU scientific library, http ://www.gnu.org/software/gsl, 2010.

D. Gu. Distributed em algorithm for gaussian mixtures in sensor networks. *IEEE Trans. Neural Netw.*, 19(7) :1154–1166, 2008.

I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S. Solla. Structural risk minimization for character recognition. *Advances in Neural Information Processing Systems*, 4 : 471–479, 1992.

P. Haase, D. Herzig, M. Musen, and T. Tran. Semantic wiki search. *Lecture Notes in Computer Science*, 5554/2009 :445–460, 2009.

L. O. Hall, I. B. Özyurt, and J. C. Bezdek. Clustering with a genetically optimized approach. *IEEE Transactions on Evolutionary Computation*, 3(2) :103–112, 1999.

J. R. Hershey and P. A. Olsen. Approximating the kullback leibler divergence between gaussian mixture models. *Proceedings of IEEE Conference on Acoustics, Speech and Signal Processing*, 2007.

J. H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, 1975.

K. Hotta. Scene classification using local co-occurrence feature in subspace obtained by kpca of local blob visual words. *Proceedings of IAPR/IEEE International Conference on Pattern Recognition (ICPR'10)*, 2010.

D. F. Huyhn and D. R. Karger. Parallax and companion : Set-based browsing for the data web. *WWW*, 2009.

K. Inoue, T. Saito, K. Shinoda, and S. Furui. High-level feature extraction using sift gmms and audio models. *Proceedings of IAPR/IEEE International Conference on Pattern Recognition (ICPR'10)*, 2010.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering : A review. *ACM Computing Surveys*, 1999.

E. Januzaj, H.-P. Kriegel, and M. Pfeifle. Towards effective and efficient distributed clustering. *Workshop on Clustering Large Data Sets, ICDM*, pages 49–58, 2003.

H. Jeffreys. *Theory of Probability.* Oxford University Press, 3rd edition, 1961.

H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. on Pattern Analysis Machine Intelligence*, 2010.

T. Joachims. Transductive support vector machines. In *Semi-Supervised Learning*. MIT Press, 2006.

D. R. Jones and M. A. Beltramo. Solving partitioning problems with genetic algorithms. *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 442–449, 1991.

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2) :181–214, 1994.

M. I. Jordan, Z. Ghahramani, Jaakkola T. S., and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37 :183–233, 1999.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, Series D(82) :35–45, 1960.

T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1) :7–15, 1989. ISSN 0020-0190.

Hyunmo Kang, Lise Getoor, and Lisa Singh. Visual analysis of dynamic group membership in temporal social networks. *SIGKDD Explor. Newsl.*, 9(2) :13–21, 2007. ISSN 1931-0145.

R. E. Kass and A. E. Raftery. Bayes factors and model uncertainty. Technical Report 254, University of Washington Seattle - Department of Statistics, 1993.

D. Keim and H.-P. Kriegel. Visualization techniques for mining large databases : a comparison. *IEEE Transactions on Knowledge and Data Engineering, Special Issue on Data Mining*, 8(6) :923–938, 1996.

A. Kermarrec. Challenges in personalizing and decentralizing the web : An overview of GOSSPLE. *Lecture Notes in Computer Science*, 5873 :1–16, 2009.

J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3) :226–239, 1998.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 2(12) :1137–1143, 1995.

T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9) :1464–1480, 1990.

B. Korte and J. Vygen. *Combinatorial Optimization : Theory and Algorithms*. Springer, 2002.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2 :83–97, 1955.

P. Kuntz, P. Layzell, and D. Snyers. A colony of ant-like agents for partitioning in vlsi technology. *Proceedings of the Fourth European Conference on Artificial Life*, pages 417–424, 1997.

N. Labroche, N. Monmarché, and G. Venturini. A new clustering algorithm based on the chemical recognition system of ants. *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 345–349, 2002.

J. Lavergne, H. Azzag, G. Venturini, and C. Guinot. Une approche incrémentale d'une méthode de classification non supervisée par nuages d'insectes volants. *XIVe Rencontre de la Société Francophone de Classification (SFC'2007)*, 2007.

M. H. C. Law, A. Topchy, and A. K. Jain. Clustering with soft and group constraints. *Lecture Notes in Computer Science*, 2004.

M. H. C. Law, A. Topchy, and A. K. Jain. Model-based clustering with probabilistic constraints. *Proceedings of SIAM Data Mining*, 2005.

L.-J. Li and L. Fei-Fei. Optimol : Automatic online picture collection via incremental model learning. *International Journal of Computer Vision*, 88(2) :147–168, 2010.

J. Liu, J. Yang, Y. Zhang, and X. He. Action recognition by multiple features and hyper-sphere multi-class svm. *Proceedings of IAPR/IEEE International Conference on Pattern Recognition (ICPR'10)*, 2010.

S. P. Lloyd. Least squares quantization in pcm. *IEEE Trans. on Information Theory*, 28(2) :129–136, 1982.

Eloïse Loubier, Wahiba Bahsoun, and Bernard Dousset. Visualization and analysis of large graphs. In *PIKM '07 : Proceedings of the ACM first Ph.D. workshop in CIKM*, pages 41–48, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-832-9.

D. G. Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, 2 :1150, 1999.

E. D. Lumer and B. Faieta. Diversity and adaptation in populations of clustering ants. *Proceeding of the Third International Conference on Simulation of Adaptive Behavior*, pages 501–508, 1994.

D.J.C. MacKay. Probable networks and plausible predictions — a review of practical bayesian methods for supervised neural networks. *Network : Computation in Neural Systems*, 6 :469–505, 1995.

T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. "neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Trans. Neural Netw.*, 4(4), 1993.

D. S. McCrickard and C. M. Kehoe. Visualizing search results using sqwid. In *In Proceedings of the Sixth International World Wide Web Conference*, pages 51–60. ACM Press, 1997.

G. J. McLachlan and Peel D. *Finite Mixture Models*. John Wiley and Sons, 2000.

J. McQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

S. Merugu and J. Ghosh. Privacy-preserving distributed clustering using generative models. *IEEE International Conference on Data Mining*, pages 211–218, 2003.

K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. *Proceedings of Eighth International Conference on Computer Vision*, 1 :525, 2001.

D. J. Miller and H. S. Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. *Neural Information Processing Systems*, 1997.

T. P. Minka. Automatic choice of dimensionality for pca. *Neural Information Processing Systems*, 2000.

N. Monmarché, M. Slimane, and G. Venturini. On improving clustering in numerical databases with artificial ants. *5th European Conference on Artificial Life (ECAL'99), Lecture Notes in Artificial Intelligence*, 1674 :626–635, 1999.

F. Murtagh. A survey of recent advances in hierarchical clustering algorithms which use cluster centers. *The Computer Journal*, 26 :354–359, 1984.

O. Nasraoui and E. Leon. On fitness, niching strategies, and hybrid niche size estimation for discovering an unknown number of clusters in noisy data. *GECCO 2004 Workshop Proceedings*, 2004.

O. Nasraoui, F. A. González, C. Cardona, C. Rojas, and D. Dasgupta. A scalable artificial immune system model for dynamic unsupervised learning. *GECCO, Lecture Notes in Computer Science*, 2723 :219–230, 2003.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models*, 1999.

S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-100). Technical report cucs-006-96, Columbia University, Department of Computer Science, 1996.

F. Nielsen and S. Boltz. The burbea-rao and bhattacharyya centroids. *arXiv*, arXiv :1004.5049v1, 2010.

F. Nielsen and R. Nock. Sided and symmetrized bregman centroids. *IEEE Transactions on Information Theory*, 55(6) :2882–2904, 2009.

K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39 :103–134, 2000.

K. Nigam, A. McCallum, and T. Mitchell. Semi-supervised text classification using em. In *Semi-Supervised Learning*. MIT Press, 2006.

S. Nikitidis, N. Nikolaidis, and I. Pitas. Incremental training of multiclass support vector machines. *Proceedings of IAPR/IEEE International Conference on Pattern Recognition (ICPR'10)*, 2010.

A. Nikseresht and M. Gelgon. Gossip-based computation of a Gaussian mixture model for distributed multimedia indexing. *IEEE Transactions on Multimedia*, 3 :385–392, March 2008.

R. Nowak. Distributed EM algorithms for density estimation and clustering in sensor networks. *IEEE Trans. on Signal Processing*, 51(8), August 2003.

A. Oliva and P. G. Schyns. Coarse blobs or fine edges ? evidence that information diagnosticity changes the perception of complex visual stimuli. *Cognitive Psychology*, 34 :72–107, 1997.

C. Ono, M. Kurokawa, Y. Motomura, and H. Asoh. A context-aware movie preference model using a bayesian network for recommendation and promotion. *Lecture Notes in Computer Science*, 4511/2009 :247–257, 2009.

T. O'Reilly. What is web 2.0. *O'Reilly Media, http ://oreilly. com/pub/a/web2/archive/what-is-web-20.html ?page=2*, 2005.

F. Picarougne, H. Azzag, G. Venturini, and C. Guinot. A new approach of data clustering using a flock of agents. *Evolutionary Computation*, 15(3) :345–367, 2007.

J. Ponce, M. Hebert, C. Schmid, and A. Zisserman. *Towards category-level object recognition*. Springer, 2006.

S. Poullot, O. Buisson, and M. Crucianu. Z-grid-based probabilistic retrieval for scaling up content-based copy detection. *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 348–355, 2007.

G. Proctor and C. Winter. Information flocking : Data visualisation in virtual worlds using emergent behaviours. *Proceedings of the First International Conference on Virtual Worlds*, 1434 :168–176, 1998.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1) :81–106, 1986.

J. R. Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, 1993.

C. E. Rasmussen. The infinite Gaussian mixture model. *Advances in Neural Information Processing Systems - MIT Press*, 12 :554–560, 2000.

I. Rechenberg. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment Translation No. 1122*, 1965.

C. W. Reynolds. Flocks, herds, and schools : A distributed behavioral model. *Computer Graphics (SIGGRAPH '87 Conference Proceedings)*, 21(4) :25–34, 1987.

D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10 :19–41, 2000.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500) :2323–2326, 2000.

Rproject. The R project, http ://www.r-project.org, 2010.

A. Runnalls. A kullback-leibler approach to gaussian mixture reduction. *IEEE Trans. on Aerospace and Electronic Systems*, 2006.

B. Safarinejadian, M. Menhaj, and M. Karrari. Distributed variational bayesian algorithms for gaussian mixtures in sensor network. *Signal Processing*, 90(4) :1197–1208, 2010.

M. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7) :1649–1681, 2001.

G. Schaefer and M. Stich. UCID - an Uncompressed Colour Image Database. *Proceedings SPIE, Storage and Retrieval Methods and Applications to Multimedia*, pages 472–480, 2004.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6 :461–464, 1978.

P. G. Schyns and A. Oliva. From blobs to boundary edges : evidence for time- and spatial-scale dependent scene recognition. *Psychological Science*, 5 :195–200, 1994.

T. Seidl, I. Assent, P. Kranen, R. Krieger, and J. Herrmann. Indexing density models for incremental learning and anytime classification on data streams. *Proceedings of the 12th International Conference on Extending Database Technology : Advances in Database Technology*, pages 311–322, 2009.

V. Smidl and A. Quinn. *The Variational Bayes Method in Signal Processing*. Springer, 2006.

A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3 :583–617, 2003.

V. Sulic, J. Pers, M. Kristan, and S. Kovacic. Dimensionality reduction for distributed vision systems using random projections. *Proceedings of IAPR/IEEE International Conference on Pattern Recognition (ICPR'10)*, 2010.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500) :2319–2323, 2000.

J. Timmis, M. Neal, and J. Hunt. An artificial immune system for data analysis. *Biosystems*, 55(1) :143–150, 2000.

M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 2001.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society - B Series*, 61(3) :611–622, 1999a.

M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2) :443–482, 1999b.

D. M. Titterington. Recursive parameter estimation using incomplete data. *Journal of the Royal Statistical Society - B Series (Methodological)*, 46(2) :257–267, 1984.

A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

UCI. Uci machine learning repository, http ://mlearn.ics.uci.edu/mlrepository.html, 2007.

N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. Split and merge em algorithm for improving gaussian mixture density estimates. *The Journal of VLSI Signal Processing*, 26(1-2) :133–140, 2000.

A. Vailaya, M. A. T. Figueiredo, A. K. Jain, and H. J. Zhang. Image classification for content-based indexing. *IEEE transactions on image processing*, 10(1) :117–130, 2001.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

N. Vasconcelos. Image indexing with mixture hierarchies. *Proceedings of IEEE Conference in Computer Vision and Pattern Recognition*, 1 :3–10, 2001.

N. Vasconcelos and A. Lippman. Learning mixture hierarchies. *Advances in Neural Information Processing Systems - MIT PressNeural Information Processing Systems*, II :606–612, 1998.

G. Von Laszewski. Intelligent structural operators for the k-way graph par- titioning problem. *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

Y. Wang, S. Luo, M. T. Freedman, and S. Y. Kung. Probabilistic principal component subspaces : A hierarchical finite mixture model for data visualization. *IEEE Transactions on Neural Networks*, 11(3) :625–636, 2000.

J. H. Jr. Ward. Hierarchical grouping to optimize an objective function. *J. Am. Stat. Assoc.*, 58 :236–244, 1963.

K. Watanabe, S. Akaho, and S. Omachi. Variational bayesian mixture model on a subspace of exponential family distributions. *IEEE Trans. Neural Netw.*, 20(11) : 1783–1796, 2009.

L. R. Welch. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4), 2003.

C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. *Advances in Neural Information Processing Systems - MIT Press*, 8 :514–520, 1996.

P. C. Wong and R. D. Bergeron. 30 years of multidimensional multivariate visualization. *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 3–33, 1997.

L. Wu, S. Luo, and X. Zheng. Integrating ilsr to bag-of-visual words model based on sparse codes of sift features representations. *Proceedings of IAPR/IEEE International Conference on Pattern Recognition (ICPR'10)*, 2010.

G. Wyszecki and W. Stiles. *Color Science : Concepts and methods, Quantitative Data and Formulae*. John Wiley and Sons, 1982.

T. Xiang and S. Gong. Model selection for unsupervised learning of visual context. *International Journal of Computer Vision*, 69(2) :181–201, 2006.

T. Zhang, R. Ramakrishnan, and M. Livny. Birch : An efficient data clustering method for very large databases. *International Conference on Management of Data Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 103–114, 1996.

Table des figures

Liste des tableaux

Liste des Algorithmes

Annexes

Annexe A
Réalisations logicielles

Les prototypes réalisés dans le cadre des chapitres 3 et 4 ont été programmés sous R [Rproject, 2010], soit avec le langage natif de R, soit sous forme de modules complémentaires programmés en C ou en C++. Les nombreuses fonctions mathématiques nécessaires pour l'implémentation d'algorithmes tels que VBGMM ou VBMPPCA ont été obtenues en intégrant la bibliothèque GSL [GSL, 2010] dans le code en C des modules appelés par R. Des paramétrages spécifiques de GCC ont été rendus nécessaires pour réaliser l'intégration conjointe des API de R et de GSL.

Souvent, au démarrage du développement, un premier prototype (ou *proof-of-concept*) a été réalisé, afin de vérifier la validité algorithmique de la méthode. Seulement, les implémentations utilisant purement R sont très lentes (surtout du point de vue de la pile d'appels, lors de l'usage de boucles ou de fonctions imbriquées). Aussi, pour obtenir des résultats plus significatifs, le portage du module préliminaire en C était réalisé.

Outre les modules de VBGMM, VBMPPCA et dérivés (à titre indicatif, la taille d'un tel module, en C, excède souvent les 1000 lignes), de nombreux utilitaires ont été utiles. Par exemple, nous pouvons citer des fonctions pour calculer des estimations de divergences KL entre mélanges de gaussiennes suivant différentes méthodes d'approximation, ou encore un module pour réaliser l'orthogonalisation de Gram-Schmidt.

Les visuels de nuages de points, et de courbes de résultats, ont été réalisés avec le système de génération graphique de R.

Les variantes de FClust ont été réalisés avec Visual Studio, en C++. La couche graphique utilise OpenGL, SDL est employée pour l'affichage des images, et freetype2 pour la génération de texte. Le système de navigation dans une collection d'image a combiné ces deux environnements : la couche visuelle est réalisée via Visual Studio, la couche classification avec R et C, et un protocole de communication a été défini afin que ces deux couches communiquent par sockets.