Estimation de modèles de mélange probabilistes: une proposition pour un fonctionnement réparti et décentralise

Afshin Nikseresht

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

ÉCOLE DOCTORALE STIM

■ SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DES MATÉRIAUX ■

Année 2008

Estimation de modèles de mélange probabilistes: une proposition pour un fonctionnement réparti et décentralisé

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE NANTES

Discipline : Informatique

présentée et soutenue publiquement par

Afshin NIKSERESHT

le 22 Octobre 2008

à la Faculté des Sciences de l'Université de Nantes

devant le jury ci-dessous

Président	:	Henri NICOLAS, Professeur	Université Bordeaux 1
Rapporteurs	:	Philippe JOLY, Professeur	Université P.Sabatier, Toulouse
		Anne BOYER, Professeur	Université Nancy 2
Examinateurs	:	José MARTINEZ, Professeur	École polytechnique de l'université de Nantes
		Marc GELGON, Professeur	École polytechnique de l'université de Nantes

Directeur de thèse : Pr. Marc Gelgon
Laboratoire : LINA (Laboratoire d'Informatique de Nantes Atlantique)

N° ED 503-007

ESTIMATION DE MODÈLES DE MÉLANGE PROBABILISTES: UNE PROPOSITION POUR UN FONCTIONNEMENT RÉPARTI ET DÉCENTRALISÉ

Probability mixture model estimation: a proposal for a distributed and decentralized operation

Afshin NIKSERESHT

favet neptunus eunti

Université de Nantes

Afshin NIKSERESHT

Estimation de modèles de mélange probabilistes: une proposition pour un fonctionnement réparti et décentralisé

xiv+174 p.

Ce document a été préparé avec $\LaTeX 2_\varepsilon$ et la classe `these-IRIN` version 0.92 de l'association de jeunes chercheurs en informatique LOGIN, Université de Nantes. La classe `these-IRIN` est disponible à l'adresse :

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

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

Résumé

Cette thèse traite de l'estimation statistique distribué, avec la motivation de, et l'application à l'indexation multimédia par le contenu. Les algorithmes et les données de divers contributeurs coopéreront vers un apprentissage statistique collectif. La contribution est un arrangement pour estimer une densité de probabilité multivariable, dans le cas où cette densité prend la forme d'un modèle de mélange gaussien. Dans ce cadre, l'agrégation des modèles probabilistes de mélanges gaussiens de la même classe, mais estimés à plusieurs nœuds sur différents ensembles de données, est une nécessité typique à laquelle nous nous intéressons dans cette thèse. Les approches proposées pour la fusion de mélanges gaussiens exigent uniquement le calcul modéré à chaque nœud et peu de données de transit entre les nœuds. Ces deux propriétés sont obtenues en agrégeant des modèles via leurs (peu) paramètres plutôt que par les données multimédia. Dans la première approche, en supposant que les mélanges sont estimés indépendamment, nous propageons leurs paramètres de façon décentralisée (gossip), dans un réseau, et agrégeons les modèles à partir des nœuds reliés entre eux, pour améliorer l'estimation. Les modèles de mélange sont en fait concaténés puis réduits à un nombre approprié de composants gaussiens. Une modification de la divergence de Kullback conduit à un processus itératif pour estimer ce modèle agrégé. Afin d'apporter une amélioration, l'agrégation est réalisée par la modélisation bayésienne du problème de groupement de composant de modèle de mélange gaussien et est résolue en utilisant la méthode variationnelle, appliquée au niveau de composant. Cela permet de déterminer, par un processus simple, peu coûteux et précis, les attributions des composants qui devraient être agrégés et le nombre de composants dans le mélange après l'agrégation. Comme seulement les paramètres du modèle sont échangés sur le réseau, le calcul et la charge du réseau restent très modérés.

Mots-clés : apprentissage distribué, calcul réparti, estimation distribuée, modèles de mélanges gaussiens (GMM), indexation multimédia, méthode variationnelle de l'apprentissage bayésien

Abstract

This thesis deals with the distributed statistical estimation, with its motivation from, and application to, multimedia content-based indexing. Algorithms and data from various contributors would cooperate towards a collective statistical learning. The contribution is a scheme for estimating a multivariate probability density in the case where this density takes the form of a Gaussian mixture model (GMM). In this setting, aggregation of probabilistic Gaussian mixture models of the same class, but estimated on several nodes on different data sets, is a typical need, which we address in this thesis. The proposed approaches for fusion only requires moderate computation at each node and little data to transit between nodes. Both properties are obtained by aggregating models via their (few) parameters, rather than via multimedia data itself. In the first approach, assuming independently estimated mixtures, we propagate their parameters in a decentralized fashion (gossip) in a network, and aggregate GMMs from connected nodes, to improve estimation. Mixture models are in fact concatenated, then reduced to a suitable number of Gaussian components. A modification on Kullback divergence leads to an iterative scheme for estimating this aggregated model. As an improvement through a change of principle over the first work, aggregation is achieved through Bayesian modelling of the GMM component grouping problem and solved using a variational Bayes technique, applied at component level. This determines, through a single, low-cost yet accurate process, assignments of components that should be aggregated and the number of components in the mixture after aggregation. Because only model parameters are exchanged on the network, computational and network load remain very moderate.

Keywords: distributed learning, distributed computing, distributed estimation, Gaussian Mixture Model (GMM), multimedia indexing, variational Bayesian learning

Acknowledgements

I would like to thank my advisor Prof. Marc GELGON for not only his support and guidance throughout this research but also for his continued assistance and insight during the writing process. He helped me enormously and it is impossible for me to thank him enough for his constant help and support as a research advisor.

I would also like to thank the members of the jury for having accepted to be part of my doctoral committee. I am grateful to my friends at LINA laboratory and to all the members of the ATLAS-GRIM group which made my student life a memorable one.

I thank my parents for the inspiring education they provided me, their love and support. It would not have been possible to carry out this work without their unconditional support.

I dedicate this thesis to my precious wife, SARA. Her love, help, encouragement and endless patience with me, given without question, has enabled me to complete this work for which I thank her sincerely with all my heart.

Contents

Résumé Etendu

I Introduction

I.1 Motivation

Les volumes de données et les applications à recherche de l'information de multimédia se développe très rapidement et est devenue une tâche importante dans les domaines d'application modernes (l'imagerie médicale, le commerce électronique, la recherche de copie illégale pour les raisons de droit d'auteur, etc) et les applications à usage général (par exemple Google Image et Youtube). Le terme de multimédia comprend les textes, les images, les documents audio et vidéo. Un besoin central et classique exprimé par l'indexation basée sur le contenu des documents de multimédia, est l'attribution d'une étiquette symbolique de classe à un document ou à une partie de celui-ci, comme l'identification d'un visage, d'un locuteur ou d'une texture ou d'un événement spatio-temporel [56]. La construction d'un moteur de recherche capable d'identifier plusieurs genres de telles classes audiovisuelles, est une tâche formidable, longtemps évaluée comme peu réaliste par la communauté de vision d'informatique. Ceci rétablit actuellement comme l'un des visions les plus stimulantes pour la recherche et les applications dans le domaine [144].

La caractérisation des classes est le travail des classificateurs. La tâche générale pour former ces classificateurs est l'apprentissage supervisé qui a besoin des données étiquetées pour l'apprentissage. Ceci implique une conception soigneuse des observations des médias spécifiques à partir des données brutes. Généralement il devrait d'autant plus des attributs qu'il y a des classes pour être distingués, qui à son tour, augmente la quantité de données nécessaires et la puissance de calcul exigée. Il y a, cependant, des tendances encourageantes vers la perspective de la collecte automatique des exemples des données de l'apprentissage à grande échelle: (i) l'analyse conjointe des textes/image, qui peut être alimentée par une ressource massive en pages Web, (ii) les progrès récents du fiable apprentissage supervisé [163, 196], qui permet l'apprentissage d'une classe de cas fournies dans le désordre (par exemple, un visage dans un fond complexe). Ceci suggère que la quantité (nécessairement énorme) des données de l'apprentissage soit distribuée par nature à grande échelle et fournie par les sources indépendantes. Ce grand volume de données peut être utilisée à l'apprentissage semi-supervisé [42], qui peut gérer conjointement la classe des données étiquetées et non étiquetées dans la phase de l'apprentissage.

D'autre part, les fin-hôtes sont de plus en plus puissants et fournissent plus de services alors que les services sur l'Internet évoluent des architectures client-serveur pleinement centralisées aux architectures entièrement distribuées. Dans la décennie passée, beaucoup de systèmes distribués ont été déployés dans une large variété des environnements, à titre exemple les réseaux pair-à-pair (P2P), les grilles de calcul , les réseaux de capteur, etc. Dans une organisation P2P [123], (i) les ressources sont dynamiques : les services des données et de l'apprentissage /classification peuvent joindre ou laisser le réseau à tout moment ; (ii) un nœud est à la fois client et serveur: il peut produire certains services et employer les services qui sont sur les autres nœuds; (iii) les ressources sont agrégées : la qualité du service global est due à son aspect collectif ; (iv) le système est décentralisé: chaque contributeur peut fournir des données ou des outils d'apprentissage, sans aucune administration centrale.

Donc, nous avons de tels réseaux à grande échelle et décentralisés dans lesquels nous pouvons trouver, d'un côté une grande quantité de données étiquetées et non étiquetées de multimédia (dans le sens de l'apprentissage supervisé) et de l'autre côté de nombreux logiciels et de différents algorithmes de reconnaissance (qui sont formés en utilisant principalement des données locales et travailent au niveau local). Les deux éléments, l'information de multimédia et l'algorithmes de reconnaissance des formes sont les deux éléments principaux pour faire un apprentissage global, mais il manque de la coordination. Le travail à l'intersection de ces domaines comprend, par exemple, l'efficacité et le placement des données liées à la recherche des images similaires [126] et l'apprentissage collectif à partir de données textuelles [175]. Nous ne pouvons pas transférer des données énormes de multimédia sur le réseau pour les accumuler sur un nœud central pour former un classificateur. Ceci motive concevoir un système d'apprentissage distribué pour apprendre les données entières sans transfert de données sur le réseau.

I.2 Objectifs de la thèse

Dans cette thèse, nous considérons les outils de l'apprentissage, mis à disposition en ligne comme services, par des divers contributeurs humains sur les nœuds. Des données multimédia sont mis sur les nœuds et une partie de celles-ci est supposée être étiquetée avec sa classe, c'est-à-dire peut contribuer à l'apprentissage supervisé. Nous supposons également que la fonction de l'espace est commune à tous les nœuds dans le réseau. Chacun de ces services d'apprentissage, emploie les données multimédia locales pour former son classificateur, mais ils aimeraient coopérer pour améliorer leur classificateur. Autrement dit, les classificateurs apprendraient les uns des autres. Nous nous concentrons sur l'apprentissage statistique et supervisé (bien que la classification non supervisée soit également intéressante). Chaque service peut recevoir les données de multimédia et envoyer l'identificateur de classe correspondante (par exemple le nom de locuteur), et / ou de demande, ou de partage des données de l'apprentissage (c'est-à-dire les paires de (données, identificateur de classe)) pour l'apprentissage supervisé. Notre objectif est d'arguer un système d'apprentissage distribué sur un réseau, c'est-à-dire de profiter de l'ensemble de données sur le réseau pour l'apprentissage. De façon décentralisée, les algorithmes et les données provenant de divers contributeurs coopéreraient en vue d'un apprentissage statistique collectif. Les idées similaires sont également en cours d'examen pour l'apprentissage collectif à partir des données textuelles [175] et des réseaux de capteur [87, 135].

Une organisation appropriée pour la vision ci-dessus est une architecture pair-à-pair dans laquelle les nœuds dirigeraient un service fournissant l'apprentissage supervisé d'une classe multimédia et stockeraient probablement quelques données de l'apprentissage. Sur demande, l'on pourrait classer les données entrantes selon les meilleures connaissances actuelles de l'architecture. Comme il s'agit d'un point de vue général, nous restreignons l'étude à l'apprentissage supervisé et décentralisé d'une classe. Nous n'abordons pas ici les problèmes importants tels que la localisation des services et des données, les arrangements élaborés de déplacement de données (Examiné dans [126], pour la récupération des images similaires), le fait que les identificateurs de classe devraient être conformes à une norme, et pas à la phase de requête.

Afin de permettre une évolution flexible de l'ensemble des classes, nous favorisons une approche générative, qui caractérise la classe en fonction de l'espace, devant une approche discriminante, qui apprend directement à la distinguer des autres classes. Cette approche générative conduit à des solutions plus souples, car l'introduction des nouvelles classes au système ne nécessite pas de mise à jour de la description des classes connues. En conséquence, dans cette thèse, nous décrirons la technique pour une seule classe. Concrètement, nous estimons la densité de probabilité de classe-conditionnelle. Nous abordons le calcul distribué d'une densité de probabilité, applicable dans la récupération des documents de multimédia et dans le contexte particulier d'un système décentralisé et distribué. Nous sommes en outre se concentrer sur les cas où toutes les densités sont les modèles de mélange gaussien. Ce modèle a beaucoup de bonnes propriétés et a été largement utilisé pour modéliser nombreux de classes de multimédia [56, 80, 148]. Pour atteindre notre objectif, nous examinons l'agrégation des différents modèles de mélange gaussien liés à une classe qui sont distribués sur le réseau. La principale caractéristique de cette approche est que, afin de maintenir le réseau et la charge de calcul modéré, les nœuds participants transmettent seulement les paramètres de mélange du modèle plutôt que les vecteurs d'attribut de multimédia, car l'agrégation ne nécessite que les paramètres du modèle.

I.3 Feuille de route

Le résumé de cette thèse est organisé selon les chapitres de la thèse. Chaque section de ce résumé corresponde à celui des chapitres de la thèse. Dans la section II nous présentons une vue d'ensemble des trois éléments d'un apprentissage distribué pour les systèmes d'identification de multimédia : les techniques de l'apprentissage automatique, l'indexation de multimédia et les techniques de récupération et le calcul distribué. Nous nous concentrons sur le modèle de mélange gaussien (GMM) dans les techniques d'apprentissage automatique que nous utilisons dans cette thèse. Ensuite, nous présentons l'estimation des mélanges gaussiens comme une approche pour l'apprentissage distribué. La section III parle de notre proposition pour l'apprentissage distribué en agrégeant les modèles de mélange. Dans la section IV nous abordons le problème pour trouver le meilleur modèle par l'utilisation de l'agrégation et employons la méthode de validation croisée pour améliorer l'algorithme. La section V décrivent une solution efficace pour agréger les modèles de mélange en utilisant la variationnelle bayésienne. Nous allons conclure dans la section VI.

II État de l'art

II.1 Apprentissage automatique

L'apprentissage automatique est un domaine de l'intelligence artificielle concernant l'étude des algorithmes informatiques qui s'améliorent automatiquement par l'expérience. Dans la pratique, ceci implique la création des programmes qui optimisent un critère de performance grâce à l'analyse des données. Il concerne le développement d'algorithmes et les techniques qui permet aux ordinateurs d'«apprendre». Apprentissage automatique a un large spectre d'applications, y compris le traitement du langage naturel, les moteurs de recherches, le diagnostic médical, la bio-informatique, la détection de fraude de carte de crédit, l'analyse de marché boursier, la classification des séquences d'ADN, l'identification de la parole et de l'écriture et l'identification d'objet en vision artificielle. La performance et le calcul d'analyse des algorithmes d'apprentissage automatique est une branche de l'informatique théorique connue sous le nom de la théorie de l'apprentissage de calcul.

Algorithmes d'apprentissage automatique sont organisés en taxonomie, basés sur les résultats désirés de l'algorithme. Les types communs d'algorithme incluent : l'apprentissage supervisé où l'algorithme produit une fonction qui relie les entrées aux sorties désirées, l'apprentissage non supervisé présenté par l'algorithme utilisant seul les exemples de l'espace d'entrée et ensuite un modèle est adapté à ces observations [82], l'apprentissage semi-supervisé qui combine à la fois les exemples étiquetés et non étiquetés pour produire une fonction appropriée ou le classificateur [20], l'apprentissage renforcé dans lequel un agent explore un environnement et à la fin reçoit une récompense, ce qui peut être positive ou négative, la transduction qui est semblable à l'apprentissage supervisé, mais ne construit pas explicitement une fonction, à la place, tente de prévoir de nouvelles sorties basée sur les entrées de l'apprentissage, les sorties de l'apprentissage et les nouvelles entrées [182] et le méta apprentissage (apprendre à apprendre).

II.1.1 L'estimation statistiques

Les méthodes statistiques peuvent être employées pour récapituler ou décrire un ensemble des données. Nous pouvons reconnaître la nature probabiliste de l'information que nous cherchons à traiter, et la forme sous laquelle nous devrions exprimer le résultat. Avec un modèle statistique à disposition, on applique la théorie des probabilités et la théorie de la décision pour obtenir un algorithme. Ceci est opposé à l'emploi des données de l'apprentissage simplement pour choisir parmi les différents algorithmes ou utiliser des algorithmes heuristiques /"sens commun" pour concevoir un algorithme.

L'objectif principal de la théorie de l'apprentissage statistique est de fournir un framework pour étudier le problème de l'inférence, qui permet d'acquérir des connaissances, de prédire, de prendre des décisions ou de construire des modèles à partir d'un ensemble de données [182]. Le but de cette théorie est d'offrir une introduction à une partie de la théorie et de la méthodologie de la classification moderne et de l'apprentissage statistique. Dans ces approches, chacun a des paramètres qui peuvent être estimés par une série de méthodes.

Diverses stratégies sont utilisées pour concevoir un classificateur en reconnaissance de formes statistique, selon le type d'information disponible sur les densités de la classe conditionnelle. Certains systèmes importants de l'apprentissage statistique sont : le classificateur linéaire [53, 18], la machine à vecteur de support (SVM) [40], la machine à vecteur de pertinence (RVM) [178], les modèles de mélange, le mélange des experts [91] et l'arbre de décision [26].

A la suite, nous présentons certains thèmes qui sont importants dans les approches statistiques:

- estimation de densité de probabilité: l'estimation de la fonction de densité de probabilité (PDF) est un concept fondamental en statistiques. Les estimateurs de densité sont les outils fondamentaux pour l'extraction de l'information incluse dans les données brutes. L'estimation de densité est la construction d'une estimation, basée sur les données observées, d'une fonction de densité de probabilité inobservable. La tâche d'estimer les paramètres s'appelle l'estimation de paramètre.

- paramétrique versus non paramétrique : les modèles paramétriques sont gouvernés par un petit nombre de paramètres adaptatifs. L'une des approches les plus simples à l'estimation de densité est de représenter la densité de probabilité $p(x)$ en termes d'une fonction spécifique à partir de laquelle contient un certain nombre de paramètres réglables. Pour

appliquer ces modèles au problème de l'estimation de densité, nous avons besoin d'une procédure de détermination des valeurs appropriées des paramètres, pour un ensemble de données observées. Les valeurs des paramètres peuvent ensuite être optimisées afin de donner la meilleure adéquation aux données. Le maximum de vraisemblance, le maximum a posteriori, l'estimation prédictive, la validation croisée, le bootstrap, l'espérance-maximisation (EM) et le boosting sont les exemples pour optimiser les modèles paramétriques. Une limitation des approches paramétriques est qu'elles supposent une forme fonctionnelle spécifique pour la distribution, qui peut s'avérer être inadéquate pour une application particulière. Une approche alternative est donnée par les méthodes d'estimation non paramétrique de densité dans lesquelles la structure du modèle n'est pas précisée a priori, mais la forme de la distribution dépend typiquement de la taille de l'ensemble des données. Ces modèles contiennent encore des paramètres, mais ceux-ci contrôlent la complexité du modèle plutôt que la forme de la distribution. Ils ont besoin d'une grande quantité de données. Les histogrammes, l'estimation de densité le plus proches voisins du noyau sont des exemples des modèles non paramétriques.
Les estimateurs paramétriques sont simples mais non flexible, en revanche, les estimateurs non paramétriques sont très flexible, mais leur précision statistique diminue beaucoup si l'on inclut plusieurs variables dans le modèle. En conséquence, les chercheurs ont essayé de développer les modèles et les estimateurs qui comportent les deux objectifs, la flexibilité et la simplicité du procédure statistique. Cela a conduit à une troisième famille, appelée semiparamétrique, qui est entre paramétriques et non paramétriques. Les estimateurs semi paramétriques combinent habituellement les dispositifs des techniques paramétriques et non paramétriques donc ils ont les composantes paramétriques (dimensionnelle finie) et les composantes non paramétriques (dimensionnelle infinie). Ils surmontent le problème de la dimensionnalité en utilisant une certaine forme de réduction de dimension.

- génératif versus discriminatif: les modèles discriminatifs sont une classe de modèles utilisés pour la modélisation de la dépendance d'une variable classe non observée sur une variable observée x. Dans un framework statistique, cela se fait par la modélisation de la distribution de probabilité conditionnelle $P(y\|x)$, qui peut être utilisée pour prédire y à partir de x. Toutes les variables d'un modèle distinctif sont directement mesurables. Ils ajustent un modèle probablement non-distributionnel pour optimiser des données à une tâche spécifique, telle que la classification ou la prédiction. Par exemple, les machines à vecteur de support [40] maximisent directement la marge d'un séparateur linéaire entre deux séries de points dans un espace euclidien. Les modèles génératifs sont les approches pour manipuler les modèles non-déterministes en décrivant ou en estimant une densité de probabilité sur les variables en question. Un modèle génératif est un modèle pour la génération aléatoire des données observées, le plus souvent quelques paramètres cachés sont donnés. Une distribution de probabilité commune sur l'observation et les séquences d'étiquette est définie par ce modèle. Les modèles génératifs sont utilisés en apprentissage automatique pour la modélisation des données, directement, ou comme une étape intermédiaire à la formation d'une fonction de densité de probabilité conditionnelle. Ceci est connu comme un modèle génératif car ayant cette distribution de probabilité, nous pouvons générer les échantillons de différentes configurations du système.

II.1.2 Modèle de mélange gaussien (GMM)

L'une des approches de l'estimation de densité de probabilité est les modèles de mélange, qui consiste à trouver une approximation appropriée de la densité. Il s'agit de modéliser une distribution statistique par un mélange (ou la somme pondérée) des autres distributions. Les modèles de mélange permettent d'obtenir les modèles complexes tout en étant basés sur plusieurs fonctions de densité de probabilité. La densité de probabilité est exprimée par la somme pondérée des densités plus simples. Les densités plus simples sont les composantes ou les états du mélange. Les modèles de mélange sont une approche populaire en raison de leurs fondements statistiques. Si nous employons les composantes gaussiennes pour la modélisation, on le connaît comme le mélange de lois gaussiennes [19]. Cette forme de modèle est omniprésente dans la modélisation des données de multimédia, car elle a de nombreuses bonnes propriétés (densité de modélisation précise, bon comportement en dimension de l'espace élevée, les procédures propres pour l'estimation et la détermination de complexité du modèle). Il a été largement utilisés pour modéliser les classes audio [148], les images [80] ou d'événement spatio-temporel dans les vidéo [56]. Ce modèle est une solution semi-paramétrique. Il est une superposition linéaire simple des composantes gaussiennes, destiné à fournir une classe plus riche des modèles de densité comparant avec le modèle gaussien simple. Dans GMM, N vecteurs d'entrée en R^d nous est donnés. Afin d'estimer la densité de probabilité de l'entrée, il utilise une combinaison linéaire des fonctions de base de K:

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

$p(x)$ est la densité au point x; $p(x/k)$ est le k-th composante de la densité; $p(k)$s sont les coefficients de mélange, c'est-à-dire la probabilité antérieure d'un vecteur de données ayant été produite du composante k du mélange. Les contraintes sont:

- $0 \leq p(k) \leq 1$
- $\sum_{k=1}^{K} p(k) = 1$
- $\int p(x|k)\, dx = 1$

Pour un grand nombre de choix de composante de fonctions de densité, le modèle de mélange peut approcher n'importe quelle densité continue avec une précision arbitraire, à condition que:

1. K est suffisamment grand,
2. Les paramètres du modèle sont choisis correctement.

Le modèle de mélange gaussien est une distribution dans laquelle la probabilité de chaque distribution gaussienne est :

$$p(x|k) = \frac{1}{((2\pi\sigma_k^2)^{d/2})} \exp \frac{\parallel (x-\mu_k)^2 \parallel}{2\sigma_k^2} \tag{2}$$

Ainsi, le mélange gaussien est défini comme une superposition de K densités de gaussien de la forme:

$$p(x) = \sum_{k=1}^{K} \pi_k N(x \mid \mu_k, \Sigma_k) \tag{3}$$

Chaque densité gaussienne $N(X \mid \mu_k, \Sigma_k)$ s'appelle une composante du mélange et a sa propre moyenne μ_k et covariance Σ_k.

Les paramètres π_k sont appelés les coefficients de mélange et sont le poids ou la densité des composantes.

Une composante k du modèle est définie par:

- **centre** μ_k: défini par la moyenne des données associée à la composante.
- **matrice de variance** Σ_k: représente la dispersion des données autour de son centre.
- **proportion de mélange** π_k: définit la fraction des données appartenant à cette composante.

Si nous intégrons les deux parties de 3 par rapport à X en notant que les deux $p(x)$ et les différentes composantes gaussienne sont normalisées, nous obtenons:

$$\sum_{k=1}^{K} \pi_k = 1 \tag{4}$$

De même, l'exigence que $p(x) \geq 0$, avec $N(X \mid \mu_k, \Sigma_k) \geq 0$, implique $\pi_k \geq 0$ pour toutes les valeurs de k. Combinant ceci avec la condition 4, nous obtenons $0 \leq \pi_k \leq 1$.

II.1.3 Estimation de paramètre en utilisant l'algorithme de l'espérance-maximisation (EM)

Une méthode puissante pour trouver la solution de maximum de vraisemblance pour les modèles avec les variables latentes est l'algorithme d'espérance-maximisation [47, 61, 134]. EM est une méthode la plus utilisée pour estimer les paramètres d'un modèle de mélange.

À partir des paramètres initiaux θ_0, l'algorithme réitère entre deux étapes :

1. **étape d'estimation:** estimer les responsabilités en utilisant les valeurs courantes du paramètre:
$$\gamma(z_{nk}) = \frac{\pi_k N(x_n \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n \mid \mu_j, \Sigma_j)} \tag{5}$$

2. **étape de maximisation:** ré-estimer les paramètres en utilisant les responsabilités actuelles. Dans le cas des matrices de variance pleines et les proportions libres de mélange, l'estimation des paramètres peut résumer par les calculs suivants:
   - les centres:
$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n \tag{6}$$
   - les matrices de variance de la forme $[\lambda_k C_k]$:
$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T \tag{7}$$

- proportions de mélange:

$$\pi_k^{new} = \frac{N_k}{N} \tag{8}$$

où

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \tag{9}$$

L'étape E correspond à l'estimation de vraisemblance de postérieur $P(x_n \mid \theta)$ à partir des paramètres θ^{m-1}. L'étape de maximisation consiste à réévaluer les paramètres θ^m. Nous devrions noter que cette phase dépend de paramétriser du modèle de mélange : le choix des contraintes sur les proportions du mélange π_k ou les matrices de variance Σ_k.

La propriété de la convergence de la probabilité a été étudiée dans [195]. Il est montré que dans les conditions générales, les valeurs de vraisemblance convergent vers une valeur stationnaire. Cette valeur est néanmoins un minimum local de probabilité. La solution optimale obtenue dépend du modèle initial θ^0. L'algorithme EM dépend donc des paramètres manuellement fixés: le nombre de composantes et des paramètres initiaux du modèle. Il est courant de lancer l'algorithme K-means afin de trouver une initialisation pour un modèle de mélange gaussien qui est ensuite adapté en utilisant EM. Les matrices de covariance peuvent commodément être initialisées aux covariances d'échantillon du cluster trouvé par l'algorithme K-means et les coefficients de mélange peuvent être réglés aux fractions de données attribuées aux clusters respectifs.

II.1.4 Techniques de combinaison de modèles

Il est souvent constaté que l'amélioration de la performance peut être obtenue en combinant plusieurs modèles en quelque sorte. Il peut être utile en particulier dans les systèmes distribués dans lesquels beaucoup de modèles peuvent exister pour une classe. On peut former L différents modèles et puis faire des prévisions en utilisant la moyenne des prévisions faites par chaque modèle. Ces combinaisons de modèles sont parfois appelées les comités (committees). Une variante importante de la méthode de comité, connue sous le nom "Boosting", implique d'entraîner plusieurs modèles dans l'ordre dans lesquels l'erreur de fonction est utilisée pour former un modèle particulier dépendant de la performance des modèles précédents. La forme la plus largement utilisée de l'algorithm de Boosting est appelée AdaBoost (Adaptive Boosting) [65]. En particulier, les points qui sont mal classés par un des classificateurs sont données plus de poids lorsqu'ils sont utilisés pour former le prochain classificateur dans la séquence. Une fois que tous les classificateurs ont été formés, leurs prévisions sont alors combinées par un arrangement de vote pondéré de majorité. Une autre manière de la combinaison des modèles est le mélange des experts. Le mélange adaptatif des experts locaux [84] est une procédure d'apprentissage qui permet une amélioration des performances dans certains problèmes en assignant différentes tâches à différents types d'apprenants. L'idée fondamentale est de former simultanément plusieurs classificateurs "experts" (ou estimateurs de régression) et une fonction de "gating". La fonction de gate assigne une probabilité basée sur l'entrée courante à chacun des experts. Elle permet aux coefficients de mélange eux-mêmes d'être les fonctions de la variable d'entrée. Le mélange de l'expert a une limitation significative à cause de l'utilisation des modèles linéaires pour la gate et les fonctions des expertes. Un modèle beaucoup plus flexible est obtenu en employant la fonction de gate multiniveau pour donner le mélange hiérarchique des experts ou le modèle de HME [90].

Dans ce modèle, les experts peuvent être construits à partir des experts de niveau inférieur et des fonctions de gates. L'architecture est un arbre dans lequel les gates de réseaux s'asseoient aux nœuds nonterminals de l'arbre.

II.2 La recherche des documents multimédia par le contenu

Les données de multimédia, telles que le texte, l'acoustique, l'images et la vidéo, sont très différentes des données structurées et des données semi-structurées (des textes) parce qu'elles sont média-spécifique (avec des opérations spécifiques), probablement très grandes, et décrites par des méta-données. Comme l'utilisation des médias numériques augmente, les techniques efficaces de récupération et de gestion deviennent plus importantes. La gestion des données de multimédia vise à fournir des capacités d'hauts niveaux pour la recherche et la manipulation de collections multimédia efficacement et précisément. Ces techniques sont nécessaires pour faciliter la capture, le stockage, la recherche et la navigation efficace dans les grandes bases de données multimédia.

La plupart des méthodes traditionnelles et communes de recherche de multimédia utilisent une certaine méthode pour ajouter des méta-données telles que la légende, les mots-clés, ou les descriptions aux documents de multimédia permettant aux médias d'être indexés en basant sur les annotations de mot-clé et sont accédés par la recherche basée sur le texte [58]. Les informations textuelles sur le multimédia peuvent être facilement recherchées en utilisant la technologie existante, mais exigent les humains de décrire personnellement chaque image dans la base de données. Cette méthode de mots-clés basée sur l'indexation a beaucoup de limitation particulièrement dans le cadre des bases de données de multimédia [50, 167].

Afin d'essayer de surmonter ces difficultés, la recherche basée sur le contenu emploie le contenu des informations pour indexer automatiquement les données et utiliser le contenu des médias plutôt que de l'annotation entrée par l'humain pour localiser données souhaitées dans les grandes bases de données [72]. Il y a des études étendues sur la conception des systèmes automatiques d'indexation et de la recherche basés sur le contenu. Pour les médias visuels ce contenu peut inclure, la couleur, la forme, la texture, le mouvement, etc. Pour les données d'acoustique/parole les contenus peuvent inclure les phonèmes, le lancement, le rythme, les coefficients cepstral, etc. Dans les systèmes typiques de recherche basés sur le contenu (CBR), le contenu des médias dans la base de données est extrait et décrit par les vecteurs d'attribut multidimensionnels, également appelés les descripteurs. Les vecteurs d'attribut des médias constituent un ensemble de données d'attribut.

Pour récupérer les données souhaitées, les utilisateurs soumettent les exemples de requête au système de recherche. Le système représente alors ces exemples avec les vecteurs d'attribut. Les distances (c'est-à-dire, les similitudes) entre les vecteurs d'attribut de l'exemple de la requête et ceux des médias dans l'ensemble de données d'attribut sont alors calculées et rangées. La récupération est conduite en appliquant un schéma d'indexation pour fournir une manière efficace de recherche dans la base de données de médias. Enfin, le système range les résultats de recherche et puis renvoie les résultats supérieurs de recherche qui sont les plus semblables aux exemples de requête.

II.2.1 La représentation du document de multimédia en utilisant la recherche de contenus (CBR)

Pour représenter les documents de multimédia, il y a généralement quatre étapes :

- extraction d'attribut : représentation des médias a besoin de considérer les attributs qui sont les plus utiles pour représenter le contenu des médias et les approches qui peuvent effectivement coder les attributs des médias. Les attributs utiles de l'objet sont extraits et transformés en vecteurs d'attribut. La base de données organise alors les vecteurs d'attribut pour la recherche basée sur le contenu. La plupart des attributs généralement utilisés pour une image incluent ceux qui se reflètent les couleur, la texture, la forme et les points saillants [71, 179, 113, 117], mais nous pouvons utiliser tout ce qui peut distinguer les classes de données. En fait, dans cette thèse on cherche uniquement des classes.

- construction des signatures à partir d'attributs: la description mathématique d'un média dans le but de la recherche est référée comme sa signature. Une fonction de densité [51] ou un modèle stochastique spatiale [105] peut être employé comme la signature.

- similitude en utilisant les signatures: après avoir trouvé les signatures, nous devons définir une manière d'évaluation de la similitude entre une paire de médias basés sur leurs descriptions abstraites. La fonction de similitude nous permet de comparer les documents de multimédia. Pour chaque type de signatures, sa représentation mathématique détermine le choix des distances et l'emploi des méthodologies relatives. Par exemple, la divergence KL, est une mesure théorétique et asymétrique de l'information de différence entre deux distributions.

- groupement et classification : un besoin important de l'indexation et de la recherche de multimédia est lié à la caractérisation des classes. La classification est en général appliquée pour annotation automatique ou pour l'organisation des documents inaperçus (unseen) en grandes catégories dans le but de la recherche. En l'absence des données marquées, le clustering non supervisé est souvent avéré utile pour accélérer la recherche et l'améliorer le résultat de visualisation.

II.2.2 Le problème de différentes descriptions de classe

Nous avons discuté la récupération basée sur contenu en utilisant des attributs. Les attributs se concentrent principalement sur le contenu des médias mais ils peuvent être employés pour concevoir les représentations de classe. Dans cette thèse, nous travaillons sur les classes au lieu des vecteurs d'attribut. On peut avoir différentes descriptions d'une classe unique. Le problème est de trouver une meilleure description des différentes descriptions existantes. Nous ne parlons pas ici de sujet de la combinaison de différents attributs pour avoir un meilleur attribut, tel que par exemple discuté dans [168] pour le cas de la recherche musicale. Nous supposons les différentes descriptions d'une classe. Ceci peut être produit quand il y a les données de l'apprentissage différentes pour une classe unique ou de différents algorithmes pour trouver les descriptions ou les modèles pour les classes. Dans ce cas, le problème est en fait la façon par laquelle l'on peut trouver la meilleure description parmi les descriptions différentes ou une meilleure description par une certaine manière de les combiner.

II.2.3 Requête

La requête est employée pour rechercher un ensemble de résultats avec le contenu semblable aux exemples spécifiques. D'après la nature de média, les requêtes dans les systèmes de recherche basé sur le contenu peuvent être conçues pour plusieurs modes (par exemple, requête par sketch, requête par la peinture [pour la vidéo et l'image], requête par le chant [pour l'acoustique], et requête par exemple). Dans le processus de requête, les utilisateurs peuvent être requis d'interagir avec le système afin de fournir la rétroaction de pertinence (relevance feedback), une technique qui permet à des utilisateurs d'évaluer les résultats de la recherche en termes de leur pertinence. La requête est concerne l'accès conceptuel aux multimédia par l'utilisateur avec un haut niveau de langage de requête. La requête commune dans le domaine du multimédia est une recherche par similarité, où les objets recherchés sont classés selon quelques points basés sur une fonction de distance définie sur un vecteur d'attribut. En présence d'indices, une requête de similitude impliquant deux ou plus d'attributs doit être décomposée en sous-requêtes dont les résultats doivent être finalement intégrés. Dans le cas des bases de données d'image, une alternative utile à la requête est la navigation. En organisant une collection d'image d'une manière appropriée, par exemple le treillis de Galois (Galois lattice), la navigation permet une recherche plus interactive et itérative.

II.2.4 Indexation

L'indexation concerne l'accès physique aux données de multimédia. L'objectif des indices est d'accéder rapidement aux données demandées par la requête. Les documents de média ne peuvent pas être recherchés sous leurs formes indigènes. Ils doivent être traités pour extraire l'information perceptuelle ou sémantique qui laissera indexer et rechercher. Les données visuelles et acoustiques sont les représentations de faible niveau et ne permettent pas les requêtes sémantiques de hauts niveaux, telles que trouver les individus dans une image ou des mots parlés dans un enregistrement audio. La récupération de média est habituellement basée non seulement sur la valeur de certains attributs, mais également sur l'emplacement d'un vecteur d'attribut dans l'espace d'attribut [63]. Les représentations perceptuelles d'image nécessitent les vecteurs de couleur, de forme, et de texture. De même, les représentations acoustiques peuvent inclure le lancement, le rythme ou la transcription de phonème. Une requête de recherche sur une base de données multimédia avec les vecteurs multidimensionnels d'attribut, exige habituellement l'exécution rapide des opérations. Les attributs de multimédia sont généralement modelés comme les points dans un espace multidimensionnel. Ainsi, le multimédia peut être efficacement indexé en utilisant les structures d'index multidimensionnelles, également appelées les méthodes d'accès spatiales. Il y a deux catégories de telles structures d'index : basée sur l'arbre (tree-based) et basée sur hachage (hashing-based).

II.3 Calcul distribué

Le calcul distribué ou répartie est un type de calcul segmenté ou parallèle. Le calcul parallèle traite le développement des programmes où plusieurs processus simultanés coopèrent à l'accomplissement d'une tâche commune. Il est le plus utilisé généralement pour se rapporter au traitement dans lequel les différentes parties d'un programme exécutent simultanément sur deux ou plusieurs processeurs qui font partie du même ordinateur. Le calcul distribué est une méthode de traitement informatique dont les différents composantes et objets comportant une application peuvent être trouvés sur différents ordinateurs connectés à un réseau de communication les uns avec les autres. Les processus séparés ne partagent pas une mémoire commune et communiquent de façon asynchrone les uns avec les autres en passant des messages sur les canaux de communication. Les ordinateurs peuvent avoir différents systèmes de fichiers et différents composantes de matériel.

Les algorithmes pour tels réseaux doivent être robustes contre les changements de la topologie. Les exemples sont les données financières rapportées sur l'Internet, les données de temps observées par un ensemble de capteurs, etc. Un des problèmes avec l'augmentation de des données distribuées est de trouver les modèles pour ces données distribuées. Le but dans cet arrangement est de concevoir les algorithmes de sorte que les calculs et les communications désirés soient fait aussi rapidement et efficacement que possible. Une approche est de recueillir toutes les données dans un nœud et de trouver le modèle des données entières. Ceci est passible pour les petites données sur un petit réseau mais pour les données de multimédia qui sont énormes et distribuées sur un grand réseau ne peut pas être fait facilement. En particulier, dans beaucoup d'applications d'exploration de données (data mining) nous sommes intéressés à l'apprentissage d'un modèle global de ces données, comme une distribution de probabilité ou un clustering des données, sans transférer d'abord toutes les données à un dépôt central. Idéalement, nous aimerions avoir un algorithme entièrement décentralisé qui calcule et dissémine des agrégats des données, avec un minimum de traitement et de communication et un bon comportement tolérance de pannes. En fait nous avons besoin d'un system de "l'apprentissage distribué" qui apprend un modèle en utilisant les données distribuées sur le réseau sans rassembler les données entières sur un nœud central.

II.3.1 Différents types de systèmes distribués

Il existe différents types de systèmes distribués, tels que les clusters, les grilles de calcul, les réseaux pair-à-pair (P2P), les systèmes de stockage distribués et ainsi de suite. Un cluster est un groupe d'ordinateurs interconnectés qui apparaissent comme un seul super-ordinateur, généralement utilisé dans la technologie scientifique en haute performance avec les applications économiques. Une grille est un type de système distribué qui permet le partage de coordonnés et l'agrégation des ressources distribuées, autonomes et hétérogènes. Les grilles sont utilisées généralement pour soutenir les applications émergés dans les domaines de l'e-science et du commerce électronique, qui font participer généralement les communautés géographiquement distribuées des personnes qui s'engagent dans les activités de collaboration pour résoudre les problèmes de large échelle et nécessitent le partage de diverses ressources telles que les ordinateurs, les données, les applications et les instruments scientifiques. Les réseaux de P2P sont les systèmes distribués décentralisés, qui permettent les applications telles que le partage de fichiers, la transmission de messages instantanée, les jeux multi-utilisateurs et en ligne et la distribution de contenu sur les réseaux publics.

II.3.2 Réseaux pair à pair

Un réseau pair-à-pair (ou P2P) se fonde principalement sur la puissance de calcul et la bande passante des participants au réseau plutôt que le concentrant dans un nombre relativement faible de serveurs. Les réseaux de P2P sont typiquement employés pour la connexion des nœuds par l'intermédiaire des raccordements en grande partie ad hoc. Ces réseaux sont utiles pour plusieurs objectifs. Le partage de fichiers contenant l'acoustique, la vidéo, les données ou quoi que ce soit dans un format numérique est très courant. Les données en temps réel, telles que le trafic téléphonique, sont également passées en utilisant la technologie de P2P. Un réseau P2P pur n'a pas la notion des clients ou des serveurs, mais seulement les nœuds égales de pair, qui fonctionnent à la fois comme «clients» et «serveurs» pour les autres nœuds sur le réseau. Ce modèle d'arrangement de réseau diffère du modèle client-serveur où la communication est généralement à et d'un serveur central et certains ordinateurs sont dédiés à servir les autres. Un exemple typique pour un transfert de fichier non P2P est un serveur FTP où le client et le serveur des programmes sont tout à fait distincts, et les clients lancent le téléchargement / téléversement et les serveurs réagissent et répondent à ces demandes. Dans le domaine de la recherche, les nœuds coopèrent afin de propager la requête (par exemple le nom du fichier) dans le réseau.

Un système de P2P est constitué d'un grand nombre de nœuds qui rejoignent ou quittent le système à tout moment. Donc la topologie de réseau ne peut être complètement connues aux nœuds du réseau et peut changer. À cette vue, les attributs peuvent maintenant se joindre au réseau depuis n'importe où avec peu d'effort; au lieu de réseaux locaux dédiés, l'Internet devient lui-même le réseau de choix. Chaque nœud est connecté à un relativement petit nombre de voisins qui, à leur tour sont connectés à plus de nœuds. Les systèmes de P2P, gagnent la popularité rapidement en raison de leur extensibilité, simplicité, l'administration simple, la tolérance aux pannes, et la nature à l'organisation autonome, ce qui porte l'espoir pour la construction des systèmes de recherche d'information à grande échelle et à faible coût [103].

Un objectif important dans les réseaux P2P est que tous les clients fournissent les ressources, y compris la bande passante, l'espace de stockage, et la puissance de calcul. La capacité et la puissance de calcul des nœuds peut être très limitées (par exemple dans le cas de réseaux de capteurs) mais, pendant que les nœuds arrivent et la demande sur le système augmente, la capacité totale du système augmente également. Ce n'est pas le cas d'une architecture client-serveur avec un ensemble fixe de serveurs, dans lesquels ajouter plus de clients pourrait signifier un transfert de données plus lent pour tous les utilisateurs. La nature distribuée des réseaux P2P augmente également la robustesse en cas d'échecs par réplication des données sur plusieurs pairs, et dans les systèmes purs de P2P en permettant aux pairs de trouver les données sans se fondant sur un serveur centralisé d'index. Dans ce dernier cas, il n'y a pas un point d'échec dans le système. Ces propriétés motivent la conception des algorithmes décentralisés simples pour le calcul où chaque nœud échange l'information avec seulement quelques-uns de ses voisins immédiats dans un temps très court.

II.3.3 Algorithmes de rumeur (gossip)

Les progrès de la technologie de réseau ad hoc, comme les réseaux P2P sur l'Internet ou les réseaux de capteurs, ont mis en évidence la nécessité des moyens efficaces d'avoir affaire avec des grands nombres de données qui sont réparties sur un ensemble de nœuds et ont nécessité la conception des algorithmes du calcul et d'échange d'information distribués et tolérante aux pannes. Une approches est d'envoyer les nouvelles données d'un nœud à d'autres nœuds jusqu'à ce que tous les nœuds aient les mêmes données. Ainsi, l'ensemble des données est mis sur tous les nœuds. Ce travail a été présenté par Demers et al. [46]. Il s'agit de l'idée d'utiliser les algorithmes épidémiques pour la mise à jour des objets de données dans une base de données repliée à beaucoup d'emplacements, par exemple, les pages jaunes, les serveurs de noms, ou les annuaires de serveur. Demers et al. proposent les deux concepts suivants :

- anti-entropie : chaque site choisit régulièrement un autre site au hasard et résout toutes les différences en échangeant le contenu complet de la base de données. Il s'avère que l'anti-entropie est extrêmement fiable mais produit énormément de communication qu'il ne peut pas être utilisé trop fréquemment.
- rumeur colportage (mongering): quand un site reçoit une nouvelle mise à jour, il devient une "rumeur à chaud". Si un site possède une "rumeur chaud", il choisit périodiquement un autre site au hasard et envoie la rumeur à l'autre site. L'idée de la rumeur colportage est d'échanger seulement les mises à jour récentes, ce qui réduit significativement les coûtes de communication. Dans la pratique, on peut utiliser une combinaison de ces deux concepts c.-à-d., en utilisant souvent les rumeurs colportage et très rarement l'anti-entropie afin de s'assurer que toutes les mises à jour sont identifiées par tous les sites.

L'idée originale pour la propagation de rumeur était d'envoyer les rumeurs seulement du l'appelant vers l'appelé (la transmission de poussée ou push transmission) [46]. Plusieurs mécanismes d'arrêt ont été étudiés pour décider quand une rumeur devient "froide" pour que son transmission soit arrêté.

Une autre idée introduite dans [46] est d'envoyer les rumeurs du appelé au appelant (transmission de traction ou pull transmission). On a observé que le nombre de sites non informés diminue beaucoup plus rapidement en utilisant un arrangement de traction au lieu d'un arrangement de poussée. Le travail de Demers et al. a lancé une énorme quantité d'étude expérimentale et conceptuelle des algorithmes épidémiques. Par exemple, il y a une série d'issues de recherches comme la cohérence, l'exactitude, les structures de données, et l'efficacité [1, 79, 112].

II.3.4 L'agrégation de nœud en utilisant l'algorithme gossip

Dans de nombreux grands systèmes, il est plus important d'avoir les valeurs agrégées du réseau entier que les données à chaque nœuds [77, 116, 181] ou en d'autres termes nous souhaitons calculer et disséminer les agrégats des données. Par agrégation nous entendons trouver les statistiques sur un ensemble de valeurs numériques qui sont distribuées, comme les valeurs, la moyenne, la somme, le compte, la variance, etc. Ceci est particulièrement nécessaire lorsque la taille des réseaux augmentent ce qui motivent pour trouver un algorithme entièrement décentralisé avec avec un minimum de traitement et de communication et un bon comportement de tolérance de pannes. Rassemblement de toutes les données locales dans un nœud spécifié rend le problème de la communication ou même le problème de stockage dans ce nœud.

Une solution est d'employer les modèles basé sur rumeurs pour le calcul [86, 96] qui fournit la robustesse, l'évolutivité et la simplicité. Dans [96] les auteurs montrent comment calculer les échantillons aléatoires ou les quantiles en utilisant une solution basée sur rumeur et décentralisée complètement. Leurs approches répondent également à beaucoup d'autres questions d'agrégation d'une mode décentralisée. Elles prolongent l'étude de l'agrégation aux sommes et aux moyennes. Cette solution peut être employée seulement pour les petites données et n'est pas appropriée pour les données énormes comme les données de multimédia. Comme la taille des données augmente, elle n'est pas facile à utiliser tels algorithmes, car la taille des données à transférer sur le réseau est énorme.

II.4 Estimation distribuée des mélanges gaussiens

Dans les sections précédentes, nous avons étudié le réseau P2P et le protocole rumeur et nous avons présenté la manière dont ils peuvent être utilisés pour le calcul distribué dans l'agrégation de nœud. L'un des problèmes avec l'augmentation des données distribuées est comment les utiliser pour apprendre les modèles de classes différentes. Ce besoin est particulièrement ressenti pour les données multimédias qui sont distribués sur tel réseaux. Une approche est de recueillir toutes les données dans un nœud et de trouver le modèle d'une classe des données entières liées à cette classe. Cet approche peut être appliquée pour les petites données ou petits réseaux, mais pour les données énormes telles que les données multimédia, ne peut pas être utilisée facilement. En fait dans beaucoup d'applications, nous sommes intéressés à l'apprentissage d'un modèle global de ces données comme une distribution de probabilité ou un regroupement des données, sans transférer d'abord toutes les données à un nœud central. Nous pouvons l'appeler "l'apprentissage distribué". Pour cela, nous avons besoin d'un algorithme décentralisé qui permet de calculer et de diffuser les agrégats des données pour trouver un modèle par le traitement et la communication minimaux.

Kowalczyk et Vlassis [99] ont proposé une implémentation décentralisée de l'algorithme EM pour apprendre un mélange de lois gaussiennes à partir de données distribuées, pour les réseaux dans lesquelles communication point à point arbitraire entre les nœuds est possible. L'algorithme appelé "Newscast EM", utilise un protocole de rumeur et tous les nœuds exécutent le même protocole en parallèle. Dans cet algorithme, on suppose un ensemble de données répartie sur les nœuds d'un réseau et tirée indépendamment un modèle de mélanges gaussiennes commun. Newscast EM utilise un protocole décentralisé de rumeur pour le calcul dans l'étape M dans un certain nombre de cycles. Le protocole est très robuste, extensible, et simple pour mettre en application. Cette approche utilise les données des différents nœuds afin de trouver un modèle. Donc, il n'est pas nécessaire d'envoyer les données sur le réseau. Ces approches peuvent être utilisées pour les données de multimédia.

Les algorithmes de type EM dépendent de l'initialisation appropriée du nombre et des paramètres de composantes gaussiennes. Pour alléger cette issue, dans [194], un algorithme d'apprentissage gourmand (greedy) et distribué est proposé pour estimer incrémentalement les composantes du mélange gaussien. Cependant, il a besoin de beaucoup de temps à apprendre le modèle utilisant cette façon et il y a toujours la possibilité d'emprisonnement dans le minimum local en raison de la nature gourmande de l'algorithme EM.

Dans la section suivante, nous proposons une approche distribuée pour l'estimation de mélanges gaussiens.

III Apprentissage distribué et peu coûteux d'un modèle probabiliste de mélange

Nous abordons le calcul distribué d'une densité de probabilité.Le calcule distribué est applicable dans la recherche documentaire de multimédia et dans le contexte particulier d'un système décentralisé et distribué.

L'objectif est de discuter d'un système d'apprentissage distribué sur un réseau P2P qui peut collaborer avec d'autres systèmes de reconnaissance des structures pour la reconnaissance du multimédia sur le réseau. De façon décentralisée, les algorithmes et les données de divers contribuants coopéreraient vers un apprentissage statistique et collectif. En fait les nœuds dans le réseau partagent leur connaissance et apprennent de plus en plus. Dans la pratique, ceci implique la communication de cette connaissance, la fusion et l'expédition de ceci. Les algorithmes incrémental tiennent compte de la fiabilité pour avoir un bon système de reconnaissance.

Nous proposons une technique pour réaliser ceci dans le cas important des mélanges gaussiens. Les modèles de mélange sont en fait concaténés puis réduits à un nombre approprié de composantes gaussiennes. Une modification sur la divergence de Kullback conduit à un processus itératif pour estimer ce modèle agrégé. La proposition implique l'envoi d'une petite quantité d'informations sur le réseau et le coût faible de calcul sur les nœuds ce qui conduit à un système très rapide. En fait, le partage de la représentation de classe est effectué par les paramètres du modèle, alors que nous n'avons pas besoin de la transmission ni du calcul de la grande quantité de données multimédia (ou les vecteurs d'attribut qui les représentent). Soulignons que la phase d'apprentissage distribué, que nous abordons dans ce travail, et la phase de requête, peut entièrement recouvrir, car la réduction de mélange maintient la représentation de classe directement prête pour l'évaluation des requêtes. Enfin, nous notons que la transmission des données pourrait être évitée d'une manière beaucoup plus simple, en transmettant les paramètres du modèle et l'échantillonnage d'eux, mais ceci implique le calcul considérable dans les espaces de grandes dimensions.

III.1 Caractéristique du système

Ici, nous parlons de la caractéristique pour concevoir le système d'apprentissage réparti de reconnaissance de multimédia.

III.1.1 Nœuds

Sur le réseau, chaque nœud peut avoir cinq partitions:

1. interface d'utilisateur: une partition qui interagit avec l'utilisateur. Elle a la capacité d'obtenir un document de multimédia de l'utilisateur pour l'envoyer comme une requête et puis donner à l'utilisateur les méta-données qu'il reçoit d'autres nœuds. Également il peut obtenir les données de multimédia et les méta-données connexes, de l'utilisateur pour les stocker dans les bases de données de son nœud.
2. partition d'index: cette partie utilise un algorithme d'apprentissage afin d'indexer les données de multimédia après avoir été entrées par l'utilisateur pour le stockage.
3. partie de récupération: elle reçoit les requêtes d'autres nœuds et envoie la meilleure réponse. Après avoir reçu la requête (audio, image, ... ou un vecteur d'attribut) elle reconnaît par l'algorithme de reconnaissance. Le système de gestion de bases de données (SGBD) renverra au demandeur les indicateurs aux méta-données qui ont été stockées pour cette classe .
4. partition de l'apprentissage: les nœuds ont besoin d'un système d'apprentissage de multimédia et de la reconnaissance pour les deux dernières partitions. La partition de l'apprentissage a le devoir de former le système de reconnaissance. Mais pour avoir un bon système de reconnaissance des structures, l'apprentissage ne peut pas être local, car il doit apprendre de plus en plus et le système devrait s'exercer avec beaucoup de données.

Dans ce système, les services de reconnaissance de multimédia sur le réseau, vont collaborer pour trouver un meilleur système de reconnaissance. Ceci se réalisera en obtenant la connaissance d'autres services de reconnaissance des structures. Ils peuvent employer le même algorithme mais formé par les meilleurs exemples. Le système peut obtenir leurs paramètres et combiner avec sa connaissance.

5. base de données: dans cette partie, trois types des données sont stockés, le multimédia, les méta-données et les paramètres de différentes classes de système de reconnaissance.
6. services : chaque nœud peut contenir certains services qui partage avec d'autres nœuds, un des services peut être le service de reconnaissance des structures de multimédia.

III.1.2 Réseau pair à pair

Il y a différents contextes de système distribué que nous pouvons utiliser dans ce problème. Cependant, pour donner les résultats généraux, nous nous efforçons de développer les solutions algorithmiques communes avec le bon niveau d'abstraction du contexte. Ainsi, nous supposons une architecture de système distribué de pair à pair (P2P) qui peut étendre aux très grandes configurations. Un modèle P2P permet des pairs de partager leurs ressources (l'information, le traitement, etc) et dans notre cas les services d'apprentissages. Une organisation pair-à-pair pour les nœuds participants semble être appropriée, car:

- le reseau est dynamique : les données et les services d'apprentissage/classification peuvent joindre ou laisser le réseau à tout moment comme les nœuds. Cela signifie que l'utilisateur peut installer un nouveau service ou désinstaller un service sur le nœud. Donc, la structure du réseau change continûment.
- un nœud est à la fois client et serveur: il pourrait apprendre à partir d'un autre nœud et / ou fournir sa connaissance à d'autres nœuds.
- des ressources sont agrégées: idéalement, la qualité du service global est due à son aspect collectif.
- le système est décentralisé: chaque contributeur peut fournir les données ou les outils d'apprentissage, sans aucune administration centrale.

III.1.3 Apprentissage supervisé distribué

Pour alléger le coût de calcul de l'apprentissage et réduire la quantité de données sur le réseau, nous examinons le cas où l'apprentissage supervisé lui-même est distribué et, plus précisément, décentralisé. En d'autres termes, nous considérons qu'une organisation prometteuse pour notre problème est une architecture pair-à-pair [123] dans laquelle les nœuds exécuteraient un service d'apprentissage supervisé d'une classe multimédia et pourraient stocker les données de l'apprentissage.

III.1.4 Apprentissage de densité de probabilité générative

L'ensemble de classes n'est pas peut-être connu a priori, dans notre scénario. (par exemple, les nouveaux locuteurs peuvent être présentés). Pour tenir compte de l'évolution flexible de l'ensemble de classes, nous favorisons une approche générative, qui caractérise la classe dans l'espace d'attribut, comparant avec une approche discriminante qui apprend le distinguer des autres. Cela simplifie le système, qui pourra alors être considéré comme autant de sous-systèmes indépendants qu'il y a des classes. Plus précisément, nous nous concentrons sur l'apprentissage d'une densité de probabilité dans un espace d'attribut. Cet espace est supposé être commun à tous les contributeurs dans le réseau, pour la classe à être apprise.

III.1.5 Modèle de mélange gaussien

Nous nous concentrons sur le cas où toutes les densités sont les modèles de mélange gaussien que nous avons discuté précédemment. Nous aimons ce modèle en raison de sa simplicité, sa polyvalence et l'utilisation omniprésente dans la littérature pour les modèles de classe de la capture audio, vidéo et image. Nous considérons un ensemble de nœuds et une certaine quantité de données de l'apprentissage étiquetée sur chaque nœud. Les modèles peuvent être estimés localement, par une technique classique. Le travail d'un apprenant de modèle de mélange est de trouver une bonne estimation du modèle. L'algorithme EM est une méthode très populaire pour trouver les paramètres d'un modèles de mélange, mais il est très coûteux.

III.1.6 calcul basé sur la méthode de rumeur (gossip)

Dans ce système nous utilisons un procédé de propagation simple de rumeur (gossip) pour fusionner les modèles. Dans les protocoles basés sur le rumeur, chaque nœud entre en contact avec un ou plusieurs nœuds à chaque tour (le plus souvent choisi au hasard), et échange de l'information avec ces nœuds. Ils permettent la propagation rapide et robuste d'une façon asynchrone et décentralisée qui adapte bien le point de vue pair-à-pair. Ils atteignent un niveau élevé de stabilité dans les conditions de stress et les perturbations.

III.2 L'apprentissage distribué pour reconnaissances des données multimédia

Il y a beaucoup de nœuds dans un réseaux P2P. Dans notre système chaque nœud peut avoir une reconnaissance des structures. Nous pourrions avoir un meilleur système de reconnaissance des structures si la connaissance des systèmes combinait l'un avec l'autre. Ainsi les nœuds peuvent apprendre entre eux en partageant leur connaissance. Les nœuds de ce système utilise le classificateur GMM et comme les classes sont représentées par un mélange de densité, elles peuvent apprendre les unes des autres en agrégeant les modèles. La première étape consiste à fusionner les modèles de mélange et à trouver un nouveau modèle. La fusion des deux GMMs produit un GMM, qui a $m_1 + m_2$ composantes où m_1 et m_2 sont les nombres de composantes gaussiennes dans les deux modèles. Mais, il n'est pas efficace d'avoir la somme des densités de deux ou plusieurs modèles. Nous devrions alors réduire la somme des densités de sorte que la distance des deux modèles soit minimale sous une certaine mesure de distance. Cela nous conduit à un problème d'optimisation. Nous présentons ici l'agrégation de mélange des modèles gaussiens et l'algorithme de l'optimisation.

La caractéristique principale de notre approche est que, afin de maintenir le réseau et la charge de calcul modéré, les nœuds participants transmettent seulement les paramètres du modèle de mélange plutôt que les vecteurs d'attribut de multimédia, puisque l'agrégation ne nécessite que les paramètres du modèle. En outre, une estimation du nombre de composantes est effectuée régulièrement, afin de permettre à plus grande échelle. Le système proposé, agrége les modèles locaux de mélanges en déterminant une combinaison appropriée de leurs composantes.

III.2.1 Agrégation légère des modèles de mélange

Supposons deux nœuds, chacun porte différents modèles probabilistics du mélange gaussien, notés $M_1(x)$ et $M_2(x)$ qui sont associés à la même entité multimédia et donc de la même densité cachée $p(x)$. Les mélanges peuvent être exprimés comme :

$$M_k(x) = \sum_{i=1}^{m_k} w_k^i N_k^i(x), \qquad k = 1,2 \tag{10}$$

où $N_k^i(x)$ est une composante gaussienne avec la moyenne μ_k^i et la covariance Σ_k^i et les w_k^i sont les poids scalaires. Le modèle M_k est estimé sur un ensemble de données de la taille n_k situées sur le nœud k. $p(x)$ peut être estimée en concaténant les mélanges comme suit:

$$M_c(x) = \frac{1}{n_1 + n_2}(n_1 \sum_{i=1}^{m_1} w_1^i N_1^i(x) + n_2 \sum_{i=1}^{m_2} w_2^i N_2^i(x)) \tag{11}$$

Cependant, $m_1 + m_2$ composantes dans M_c sont généralement en grande partie redondantes, ce qui implique une augmentation inutile de coût d'évaluation (par exemple) de probabilités pour cette densité lors de la requête, quand les fusions sont enchaînées. Par conséquent, l'élargissement de ce système nécessite la transformation de M_c en un mélange réduit M_r qui préserve assez bien la densité en ayant seulement le nombre nécessaire de composantes pour cela. Ainsi, l'ordre de grandeur du nombre de composantes est maintenu constant à travers la propagation, bien que dans le détail, il fluctue pour adapter la complexité de la densité.

III.2.2 Définition et optimisation de la similitude entre M_c et M_r

Nous voulons grouper les composantes de M_c dans un mélange réduit de $m_r < m_c$ composantes. Si nous dénotons l'ensemble de tous les modèles gaussiens de mélange avec au maximum m_r composantes par $\mathcal{M}_{m_r}$, une façon de formuler l'objectif du groupement, est de dire que nous souhaitons trouver l'élément M_r de $\mathcal{M}_{m_r}$ "plus proche" à M_c sous une certaine mesure de distance. En d'autres termes nous cherchons un modèle M_r qui maximise la probabilité prévue des données D supposées être tirées de M_c (12). L'on peut montrer [18] que ceci minimise la divergence de Kullback-Leibler $KL(M_c \| M_r)$ définie par (14) qui, en bref, mesure la perte d'informations due à l'approximation de M_c par M_r.

$$\begin{aligned}
\hat{M_r} &= \arg\max \quad E_{M_c} \; [\; ln \;\; p(D|M_r) \;\;] && (12)\\
\hat{M_r} &= \arg\min \quad \left[-\int M_c(x) \; ln \; M_r(x) \; dx \right] && (13)\\
\hat{M_r} &= \arg\min \quad \left[-\int M_c(x) \; ln \; \frac{M_r(x)}{M_c(x)} \; dx \right] && (14)
\end{aligned}$$

Un problème majeur est le manque de la forme fermée pour leur divergence, dans le cas de mélanges gaussiens. Nous contournons cette issue en recourant à une variante de cette divergence proposé par [70] consistant à minimiser la mesure de similitude suivante (15):

$$d(M_c, M_r) = \sum_{i=1}^{m_1+m_2} w_c^i \min_{j=1}^{m_r} KL(N_c^i \| N_r^j) \tag{15}$$

où N_c^i (resp. N_r^i) est le i^{me} composante de M_c (resp. de M_r).
Cette mesure de similitude présente deux bonnes propriétés:

- elle se calcule aisément et rapidement, car la divergence de Kullback entre deux gaussiennes dont les parametres (μ_1, Σ_1) et (μ_2, Σ_2), dispose de l'expression analytique simple suivante:

$$\frac{1}{2}(log\frac{|\Sigma_2|}{|\Sigma_1|} + Tr(\Sigma_2^{-1}\Sigma_1) + (\mu_1 - \mu_2)^T\Sigma_2^{-1}(\mu_1 - \mu_2) - \delta) \tag{16}$$

  où δ est la dimension de l'espace d'attributs dans lequel sont définies les densités de probabilités mises en jeu.
- elle conserve une propriété théorique proche de celle par laquelle nous avons initialement justifié l'usage de la divergence de Kullback : quand le volume de données tend vers l'infini, minimiser $d(M_c, M_r)$ revient à maximiser l'espérance de la log-vraisemblance des données (supposées tirées de M_c), conditionnellement au modèle M_r et sous l'hypothèse que toutes les données tirées d'une composante de M_c sont affectées, dans la réduction de modèle, à la même composante de M_r.

Par conséquent, en suivant [70], sous la distance 15, le modèle réduit optimal $\hat{M_r}$ est la solution à minimisation de 15 pour $\mathcal{M}_{m_r}$: $\hat{M_r} = \arg\min_{M_r} d(M_c, M_r)$. La densité optimale $\hat{M_r}$ est un modèle obtenu en groupant les composantes de M_c dans les clusters et effondrant toutes gaussiennes dans un groupe, en un seul gaussienne. Nous optimisons localement le critère (15) avec un système itératif, détaillé dans l'algorithme 1 qui pourrait très rudement être comparé à un algorithme de k-means appliqué aux composantes du mélange. En effet, dans les applications de multimédia, le nombre de composantes gaussiennes étant généralement de l'ordre de 10, plutôt que plusieurs milliers pour les vecteurs d'attribut, la convergence (dure) est obtenue rapidement.

III.2.3 Détermination du nombre de composantes

Un point important dans la démarche décrite est la nécessité de la détermination du nombre de composantes gaussiennes du mélange réduit M_r. L'étude séminale rapportée dans [3] a montré que l'estimation de la divergence de Kullback est affectée par un biais qui se développe avec le nombre de paramètres à estimer, c.-à-d. avec le nombre de composantes. Il fournit également une approximation de premier ordre de cette correction, que nous appliquons ici pour la définition de $d(M_c, M_r)$, qui devient donc:

$$d(M_c, M_r) = \sum_{i=1}^{m_1+m_2} w_i \min_{j=1}^{m_r} KL(N_c^i \| N_r^j) \quad + \quad \nu_{M_r} \tag{17}$$

où ν_{M_r} désigne le nombre de paramètres du mélange réduit. Les résultats expérimentaux correspondants ont pour l'essentiel donné les résultats conformes à ceux établis par des approches usuelles (AIC ou BIC) sur le mélange optimal estimé directement sur l'ensemble des données impliquées dans la fusion. Nous évaluons exhaustivement de 1 à $m_1 + m_2$ le nombre de composantes du mélange réduit, en phases indépendantes. Une alternative plus économe de calcul serait de procéder à une réduction récursive du modèle de $m_1 + m_2$ à 1, mais ses premières étapes pourraient réduire trop rapidement le champ d'exploration de l'espace des regroupements possibles.

III.3 Validation

Pour la validation, d'abord nous avons employé trois nœuds, chacun appris une densité de probabilité pour le locuteur A dans l'espace d'attributs, supposé commun, qui est celui des 13 premiers coefficients mel-cepstraux. Les trois nœuds fusionnent simultanément en un seul. Chaque nœud a disposé de données d'apprentissage différentes de ce même locuteur. Les enregistrements audios durent entre 7 et 16 secondes selon les nœuds, c.a.d. sont relativement courts. Les densités estimées peuvent donc varier significativement d'un nœud à l'autre. Pour assurer que la différence entre les mélanges dépend essentiellement des données et non de la technique d'estimation, nous avons estimé pour chacun des trois nœuds la densité en utilisant de l'algorithme EM, avec une stratégie de multiples optimisations préliminaires partant d'initialisations aléatoires, puis d'optimisations locales poussées pour les meilleurs résultats préliminaires. Chaque nœud a estimé de manière autonome le nombre de composantes.

Les trois nœuds d'entrée ont estimé avoir 4,4 et 5 composantes. La concaténation de ces trois mélanges donne un mélange à 13 composantes, qui sera ensuite réduit à un nombre de composantes à déterminer. La densité est estimée sur l'ensemble des données et elle est nettement mieux approximées par le mélange réduit que par n'importe quel mélanges entrants. A titre de comparaison, une détermination expérimentale du nombre de composantes calculée sur l'ensemble des données (selon un critère BIC) a fourni, comme le modèle réduit, un mélange à 4 gaussiennes.

Pour évaluer la performance de la technique proposée dans le contexte distribué, nous appliquons un algorithme simple de propagation par rumeur ('gossip'). Un réseau de 13 nœuds est utilisé dans l'expérience. Chaque nœud dispose de données différentes mais toutes les données sont issues du même locuteur. Pour évaluer l'aptitude d'un mélange à modéliser les données D issues de la classe d'intérêt (ici, un locuteur), nous retenons la vraisemblance marginalisée [114]. Les données D utilisées pour cette évaluation sont l'union des données. Le calcul pratique de la vraisemblance marginalisée est réalisé au moyen de l'approximation BIC (Bayesian Information Criterion), que l'on cherche à minimiser.

Les observations suivantes peuvent être formulées:

1. le processus se stabilise autour d'un "modèle collectif". Il n'y a néanmoins pas de convergence globale assurée, car on ne réalise pas collectivement l'optimisation d'un critère global au réseau, comme c'est le cas pour le calcul distribué de moyenne [99, 22].

2. la performance de ce modèle collectif est rapidement meilleure que celle de chacun des nœuds en situation initiale (légèrement ou largement meilleure, selon les nœuds).

3. la performance du modèle collectif est assez proche d'un modèle que l'on aurait estimé directement sur l'ensemble des données.

Enfin, il est bien entendu possible qu'un nœud rejoint le système. Dès ses premiers échanges, le nœud qui rejoint le réseau "récupère" rapidement la tendance centrale.

Algorithm 1 Algorithme itératif d'optimisation du mélange réduit M_r (critèrrion (15))

```
for $m_r$ : de 1 à m1+m2 do
  Partir d'une initialisation $\hat{\pi}^0$ (aléatoire ou donnée)
  $it = 0$
  repeat
    1.Ajustement du mélange $M_r$ :
    étant donné le regroupement $\hat{\pi}^{it}$, obtenu initialement ou calculé à l'itération précédente, on met à jour les paramètres du mélange réduit :

    $$\hat{M_r}^{it} = \arg\min_{M_r \in \mathcal{M}_{m_r}} d(M_c, M_r, \hat{\pi}^{it}) \qquad (18)$$

    où $\mathcal{M}_{m_r}$ désigne l'ensemble des modèles à $m_r$ composantes que l'on peut former par le regroupement des composantes de $M_c$. Cette ré-estimation revient en fait à mettre à jour chaque composante de $M_r$ comme suit. Pour la composante $j$, l'algèbre linéaire aboutit aux expressions suivantes :

    $$\hat{w}_r^j = \sum_{i \in \pi^{-1}(j)} w_c^i, \qquad \hat{\mu}_r^j = \frac{\sum_{i \in \pi^{-1}(j)} w_c^i \mu_c^i}{\hat{w}_r^j}, \qquad ,\hat{\Sigma}_r^j = \frac{\sum_{i \in \pi^{-1}(j)} w_c^i(\Sigma_c^i + (\mu_c^i - \hat{\mu}_r^j)(\mu_c^i - \hat{\mu}_r^j)^T)}{\hat{w}_r^j} \qquad (19)$$

    où $\pi^{-1}(j)$ désigne, de façon allégée, $\hat{\pi}^{-1,it}(j)$, ensemble des composantes de $M_c$ se projettent sur la composante $j$ de $M_r$. Notons que même si l'on choisit initialement les matrices de covariances diagonales ou sphériques aux bords du réseau (où les modèles sont estimés à partie de données), la fusion rend les matrices de covariances de forme pleine.
    2. Regroupement des composantes :
    pour le mélange $\hat{M}_r^{it}$ obtenu dans la phase précédente, on recherche l'application $\pi^{it+1}$, définie de $\{1, \ldots, m_1 + m_2\}$ dans $\{1, \ldots, m_r\}$, qui correspond au meilleur regroupement des composantes de $M_c$ pour former celles de $\hat{M}_r^{it}$, au sens du critère suivant :

    $$\hat{\pi}^{it+1} = \arg\min_{\pi} d(M_c, \hat{M}_r, \pi) \qquad (20)$$

    Autrement dit, chaque composante $i$ de $M_c$ se projette sur la composante $j$ de $\hat{M}_r^{it}$ qui est la plus proche, au sens de la divergence de Kullback ((21) ci-dessous). Dans cette phase, une exploration exhaustive de l'ensemble des composantes cibles est réalisée à faible coût de calcul, grâce à l'expression analytique(16).

    $$\pi^{it+1}(i) = \arg\min_{j} KL(N_c^i || N_r^j) \qquad (21)$$

    3.$it$=$it$+$1$
  until jusqu'à convergence (c.a.d. $\pi^{it+1} = \pi^{it}$)
  Calculer $d(M_c, \hat{M}_r) = \sum_{i=1}^{m_1+m_2} w_i \min_{j=1}^{m_r} KL(N_c^i \| N_r^j) \quad + \quad \nu_{M_r}$
end for
Retenir le modèle $\hat{M}_r$ ayant minimisé $d(M_c, \hat{M}_r)$ sur l'ensemble des complexités de modèles candidats.
```

IV Choisir les meilleurs modèles par la méthode de validation croisée

Bien que l'agrégation des modèles probabilistes de mélange de lois gaussiennes discutée dans la section précédente, peut diminuer la valeur de BIC de la plupart de GMMs qui contribue au processus, il ne garantit pas que la fusion des deux GMMs a une meilleure probabilité aux données entières par rapport aux deux GMMs initiales. D'ailleurs, parfois la valeur de BIC augmente après l'agrégation de certains des modèles. Autrement dit, la fusion des paramètres du modèle n'augmente pas toujours l'efficacité des deux mélanges qui contribuent à la fusion. C'est probablement à cause de la supposition faite dans cet algorithme selon lequel le nœud avec un plus grand nombre de données est un meilleur modèle donc il a plus d'effet sur les résultats. Mais l'on ne peut pas différer entre les bons et les mauvais modèles avant de fusionner les modèles. En fait pour certains des nœuds, les paramètres du modèle n'ont pas pu être bien estimés et ceci peut affecter les autres mélanges quand le fusionnement se produit.

Dans le nouvel algorithme, une technique a été proposée pour évaluer les modèles qui sont distribués sur le réseau. Utilisant cette technique nous pouvons comparer et trouver les meilleurs mélanges ou même trouver les nouveaux modèles qui sont meilleurs (avoir plus de vraisemblance ou moins de composantes) en fusionnant les autres modèles avec un bas coût de calcul. Pour ce nouvel algorithme nous devons distinguer les bons modèles et les mauvais modèles, afin d'éviter que les mauvaises modèles affectent le résultat lorsque nous fusionnons les modèles. En d'autres termes, nous devons trouver le modèle qui a la meilleure vraisemblance aux données entières, mais comme le réseau est grand et les données de multimédia sont énormes, il n'est pas simple de faire cela pour le réseau entier. Nous pouvons donc trouver le meilleur nœud localement sur chaque région du réseau. Nous considérons chaque nœud et ses voisinages comme une région et nous employons la méthode de validation croisée pour trouver le meilleur modèle dans chaque région. Ceci est fait en assignant des points à chaque modèle qui est calculé par la méthode de validation croisée. Ensuite nous pouvons comparer les modèles et trouver le meilleur modèle selon les autres nœuds dans cette région.

Dans cette section, nous décrivons notre nouvel algorithme qui utilise une combinaison de validation croisée et la fusion pour trouver le meilleur modèle parmi de nombreux modèles qui existent sur le réseau pour une classe unique. L'algorithme a deux parties:

- par la méthode de validation croisée, trouver le meilleur modèle d'une classe parmi plusieurs modèles disponibles pour cette classe sur une région.
- afin de créer les meilleurs modèles, nous essayons de fusionner les modèles, en concaténant les modèles et de réduire un bon nombre de composantes. Par la fusion, on peut trouver les nouveaux modèles qui ont plus de vraisemblance avec la même ou moins de complexité.

IV.1 Validation croisée

Validation croisée parfois appelée l'évaluation de rotation [98], est la méthode statistique de diviser un échantillon de données dans les sous-ensembles tels que l'analyse est d'abord effectuée sur un seul sous-ensemble, alors que les autres sous-ensembles sont maintenus pour l'usage suivant en confirmant et en validant l'analyse initiale. Le sous-ensemble initial de données est appelé l'ensemble de l'apprentissage et les autres sont appelés les ensembles de la validation ou de l'essai.

IV.2 Validation croisée par vraisemblance

Sur un réseau, il y a beaucoup de nœuds qui ont estimé leurs modèles en utilisant les données qui existent sur le nœud. Estimation peut être faite par l'algorithme de EM ou d'autres algorithmes. Certains modèles peuvent bien estimer et certains non. Le processus d'estimation peut être effectué par deux facteurs:

1. données: les données sont les facteurs les plus importants pour avoir un bon modèle. Si nous avons des petites données, ou nous n'avons pas de bons exemples pour l'estimation, le risque d'avoir un mauvais modèle sera augmenté.
2. algorithme d'estimation : le choix d'algorithme pour estimer un modèle en utilisant les données, est très important pour la création de bons modèles. Par exemple algorithme EM est très sensible à l'état initial. Il peut tomber dans le minimum local dans le processus de estimation et alors ne peut pas trouver un modèle approprié bien que nous ayons de bonnes données.

Pour améliorer l'efficacité des reconnaissances de forme sur le réseau, nous devons trouver les meilleurs modèles sur le réseau. Par le meilleur modèle nous voulons dire les modèles qui ont la vraisemblance maximale aux données entières sur le réseau. Comme les données sont distribuées sur le réseau, chaque nœud a une partie de celui-ci. Les données de multimédia sont énormes et nous ne pouvons pas les envoyer sur le réseau pour rassembler les données entières dans un nœud. Ainsi il n'est pas facile de trouver les modèles qui ont la vraisemblance maximale aux données entières.

La taille des paramètres du modèles est petite, donc nous pouvons les envoyer à faible coût sur le réseau. Pour comparer les modèles, c'est une bonne idée de trouver la vraisemblance de chaque modèle aux données d'autres nœuds en les envoyant à d'autres nœuds sur le réseau. Nous avons besoin d'un système pour scorer les modèles sur les nœuds. La validation croisée est une approche pour trouver de meilleurs modèles. Nous pouvons utiliser ce moyen pour vérifier le modèle de chaque nœud avec les données des autres nœuds. Chaque nœud et ses voisins sont considérés comme une région. Dans cette configuration chaque région a l'intersection avec d'autres régions. Dans une région avec K nœuds, il y a K ensemble de données et K modèles. Nous employons la validation croisée K-fold pour examiner les modèles. Chaque nœud compare son modèle à d'autres $K-1$ ensemble de données qui existent sur la région. Chaque nœud dans la région envoie ses paramètres du modèle à tous les nœuds dans cette région et ils calculent la valeur de la vraisemblance du modèle reçue à leur données. Si le nouveau modèle produit plus de vraisemblance aux données du nœud (comparaison avec son modèle) le score pour le modèle envoyé sera augmenté.

En fait les points de chaque modèle montrent le nombre de nœuds sur la region dans lequelle ce modèle produit une meilleure vraisemblance que ses propres modèles. Les points les plus hauts montrent les meilleurs modèles. Il est possible d'avoir quelques nœuds avec le même score. Pour trouver le meilleur modèle parmi les modèles qui ont tous le meilleure score, nous utilisons la somme des vraisemblances de chaque modèle à tous les ensembles de données dans la région et le modèle avec une vraisemblance supérieure sera choisi.

Maintenant, nous résumons notre algorithme de validation-croisée : nous définissons deux états pour les nœuds: état actif et état passif. Quand un nœud envoie son modèle aux autres nœuds il est dans l'état actif et quand il reçoit le modèle d'autres nœuds il est dans l'état passif. Les algorithmes pour ces deux états sont respectivement indiqués dans les algorithmes 2 et 3.

Algorithm 2 Algorithme de validation croisée sur une région pour un nœud actif

```
1. Trouver les autres nœuds dans la région qui ont été choisis au hasard. S'il y a $K$ nœuds
dans la région, définir le nombre de 1 à $K - 1$ à eux.
for $i$ : de 1 à $K - 1$ do
  a. envoyer les paramètres du modèle au nœud $i$
  if le point est retourné depuis le nœud $i$ est égale à 1 then
    ajouter 1 au score de ce modèle
  end if
end for
2. Envoyer la valeur de score aux autres nœuds.
3. Trouver le modèle avec le maximum de score (le meilleur modèle).
4.(Pour les nœuds qui ont le même point de maximum)
if il y a plus d'un modèle qui a la valeur maximale pour le score then
  Trouver le modèle qui a le maximum de vraisemblance aux données entières sur la région
  (le même algorithme mais en utilisant la vraisemblance comme score)
end if
4. (Uniquement pour le nœud contenant le meilleur modèle) envoyer le meilleur modèle local
à tous les nœuds dans la région.
```

Algorithm 3 Algorithme de validation croisée sur une région pour un nœud passif

```
1. Recevoir les paramètres du modèle.
2. Trouver le $L_1$ = vraisemblance du modèle reçu.
3.Trouver le $L_2$ = vraisemblance de son propre modèle.
if $L_1 > L_2$ then
  définir S = 1 (score pour le nouveau modèle).
else
  définir S = 0.
end if
4. Retourner S (points pour le modèle reçu) et $L_1$ (valeurs de probabilité du modèle reçu) au
nœud d'expéditeur.
```

IV.3 Amélioration des modèles par l'agrégation

Les algorithmes décrits précédemment, peuvent trouver un meilleur modèle local dans la région selon les nœuds dans la région (voisinages d'un nœud). Après avoir trouvé le meilleur modèle au niveau local, la question est de trouver les meilleurs modèles par la fusion des modèles de cette région. En théorie, on peut d'abord concaténer les composantes de tous les nœuds puis les réduire. Nous mentionnons deux problèmes pour cela:

1. les expériences ont montré que le risque de tomber dans un minimum local sera augmenté.
2. les mauvais modèles peuvent affecter le résultat.

Nous avons donc décidé de concaténer-réduire les modèles utilisant le rumeur (gossip). Ainsi, deux nœuds seront choisis au hasard dans la région et leurs composantes seront concaténées. Comme nous n'aimons pas augmenter la complexité du modèle, nous limitons le nombre de composantes en phase de réduction au nombre de composantes du meilleur modèle trouvé dans la phase précédente. Autrement dit, pour trouver les nouveaux modèles, le mélange concaténés

Algorithm 4 La validation croisée avec l'algorithme de fusion dans une région

```
1. Trouver le modèle qui a la meilleure valeur de vraisemblance locale pour les données entières
de cette région en utilisant la validation croisée comme algorithme 2.
2. définir $K$ = Nombre de nœuds dans la région.
for $i$ : de 1 à $K^2$ do
  a. choisir au hasard deux nœuds de la région.
  b. concaténer les modèles des deux nœuds sélectionnés.
  for $j$ : de 1 au nombre de composantes dans le précédent meilleur modèle local do
    Réduire le modèle concaténés à $j$ composantes.
    if $\sum_{i=1}^{K}(log(p(NewModel|\theta_i))) \geq \sum_{i=1}^{K}(log(p(OldModel|\theta_i)))$
    et $\nu_{NewModel} \leq \nu_{OldModel}$
    (le nouveau modèle est meilleur que le précédent modèle local) then
      I. obtenir le nouveau modèle comme le meilleur modèle local.
      II. remplacer les modèles des deux nœuds par ce nouveau modèle.
    end if
  end for
end for
4. Envoyer le meilleur modèle local à tous les nœuds dans la région.
```

sera réduit à tous les nombres de composantes plus petits que le nombre de composantes du meilleur modèle dans la phase précédente.

Pour comparer ces nouveaux modèles avec l'ancien meilleur modèle local, nous avons besoin d'un critère. Le critère AIC est défini comme suit:

$$AIC(D|M) = -2logp(D|\theta) + 2\nu \tag{22}$$

où le mélange M est défini par un vecteur de paramètres θ et $p(D|\theta)$ est la vraisemblance aux données pour ce modèle, ν est le nombre de paramètres indépendants dans le mélange. Dans notre cas, nous devons trouver la vraisemblance d'un nouveau modèle à tous les ensembles de données sur la région et nous n'aimons pas que la vraisemblance soit diminuée. Donc nous avons employé les conditions suivantes pour choisir le nouveau meilleur modèle au niveau local:

$$\sum_{i=1}^{K}(log(p(NewModel|\theta_i))) \geq \sum_{i=1}^{K}(log(p(OldModel|\theta_i))), \tag{23}$$

$$\nu_{NewModel} \leq \nu_{OldModel} \tag{24}$$

où K est le nombre de nœuds dans la région. Autrement dit, le nouveau modèle sera choisi dans deux cas:

1. sa vraisemblance à toutes les données de la région est mieux que le précédent meilleur modèle au niveau local et il n'est pas plus compliqué que le modèle précédent.
2. le nouveau modèle a la même valeur de vraisemblance que le meilleur modèle local, mais a moins de complexité.

Le processus de la fusion est répété plusieurs fois pour s'assurer que les modèles dans la région ne varient pas beaucoup par la nouvelle fusion. Dans nos expériences nous définissons le nombre de la fusions, à la puissance de deux du nombre de nœuds dans la région. Maintenant nous pouvons ajouter la fusion à l'algorithme 2 et avoir l'algorithme 4.

Il faut noter qu'à la fin du processus dans une région, tous les nœuds ont le même modèle qui est choisi comme le meilleur modèle dans cette région. Comme les régions ont l'intersection avec d'autres régions, les meilleurs modèles de différentes région concurrencent et distribuent sur le réseau.

IV.4 Validation

Pour la validation, l'exemple de reconnaissance distribuée du locuteur est pris, mais la technique s'applique directement à une large gamme de classes multimédia. Dans toutes les expériences, tous les nœuds ont appris une densité de probabilité pour le locuteur 'A' dans un espace d'attribut commun de mels-cepstral de dimension 1. Un réseau de 15 nœuds est employé dans cette expérience. Chaque nœud est relié au réseau par un certain nombre de nœuds. Pour chaque nœud différentes données de l'apprentissage du même locuteur sont fournies et la durée des enregistrements audio était environ 10 secondes et chaque nœud devrait fournir son mélange estimé à partir de ses données. À cet effet, l'algorithme d'optimisation local d'espérance-maximisation est utilisé, mais avec une certaine amélioration [16] pour limiter les minimums locaux. Chaque mélange détermine automatiquement le nombre de composantes.

Pour comparer l'efficacité du modèle sur les données entières sur le réseau, nous avons choisi deux paramètres : 1) la vraisemblance à l'ensemble des données 2) le nombre de composantes des modèles. Nous avons éffectué trois expériences. Chaque expérience a été répétée 10 fois.

Dans la première expérience, nous avons trouvé chaque meilleur modèle local dans chaque région seulement par la fusion des modèles existants dans cette région. Dans la région choisie, les modèles ont été choisis par la méthode de rumeur, fusionnés et réduits à un nouveau modèle. Les expériences montrent que la valeur de vraisemblance est un peu mieux que le modèle direct. Aussi, le nombre moyen de composantes est plus faible par rapport à modèle direct, ce qui signifie qu'il a la complexité inférieure comparé au modèle direct.

Dans la deuxième expérience nous avons utilisé seulement la validation croisée pour trouver le meilleur modèle local. Tous les nœuds dans toutes les expériences (répétition 10) ont trouvé le meilleur modèle (le modèle qui a la meilleure vraisemblance aux données entières) mais ce modèle a beaucoup de composantes et donc est plus complexe.

Nous avons ajouté la fusion à l'algorithme de validation croisée dans la troisième expérience et nous avons essayé de trouver les meilleurs modèles dans cette région. L'algorithme pourrait trouver le modèle avec plus de vraisemblance et moins de complexité par rapport à l'algorithme de validation-croisée.

V Apprentissage décentralisés d'un mélange gaussien en utilisant l'agrégation variationnelle basée sur Bayes

Dans les sections précédentes, nous avons présenté une technique distribuée d'estimation statistique, avec l'application à, l'indexation basée sur contenu de multimédia. Cette contribution a été un système d'estimation d'une densité multivariable de probabilité, dans le cas où cette densité prend la forme d'un modèle de mélange gaussien. Ce dernier a une large application dans la modélisation d'attribut de multimédia. Supposant indépendamment les mélanges estimés, nous propageons leurs paramètres d'une mode décentralisée (rumeur) dans un réseau, et agrégeons les GMMs à partir des nœuds connectés, afin d'améliorer l'estimation. Dans cette section, afin d'apporter une amélioration par un changement de principe sur les travaux antérieurs, l'agrégation est réalisée par la modélisation bayésienne du problème de groupement de composante de GMM et résolu en utilisant une technique variationnelle de Bayes, appliquée au niveau des composantes. Ceci détermine, par un processus simple, peu coûteux mais précis, les attributions des composantes qui devraient être agrégées et le nombre de composantes dans le mélange après l'agrégation. Puisque seulement les paramètres du modèle sont échangés sur le réseau, la charge du calcule et du réseau reste très modérée.

V.1 Une revue sur les méthodes bayésiennes variationnelles

Une méthode standarde pour apprendre un modèle est le maximum de vraisemblance (ML). Cette méthode estime les valeurs optimales pour les paramètres du modèle dans une structure graphique fixe à partir d'une série de données. Il y a trois problèmes principaux liés à l'apprentissage ML:

1. il produite un modèle qui a le problème de surapprentissage des données et par conséquent a la performance sous-optimale.
2. il ne peut pas être utilisé pour apprentissage de la structure du graphique car les graphes plus compliqués assignent une vraisemblance plus élevée aux données.
3. il est résoluble (tractable) seulement pour une petite classe de modèles.

V.1.1 Apprentissage Baysien ou l'inférence Bayésienne

Le framework bayésien fournit, en principe, une solution pour les deux premiers problèmes. Dans ce framework on considère un ensemble de modèles, caractérisé par une distribution de probabilité sur toutes les valeurs des paramètres et des structures possibles. L'incertitude de modèle est donc prise en compte, ce qui permet d'améliorer les performances de généralisation. En outre, les modèles complexes sont effectivement pénalisés en étant assignés une probabilité postérieure plus faible, par conséquent la structure optimale peut être identifiée.

L'avantage principal de l'inférence bayésienne est que l'incertitude sur les paramètres du modèle est tenue compte et que cette approche permet de déterminer la complexité de modèle optimal sans devoir recourir à des techniques de ré-échantillonnage statistique. Apprentissage bayésien se compose de deux étapes : ajustage de modèle et choix de modèle. Pendant l'ajustage de modèle, on suppose que la structure du modèle M est fixée. Les paramètres sont appris compte tenu des données observées X. L'application de la règle de Bayes nous permet de mettre à jour notre croyance antérieure (prior) sur Θ à une croyance postérieure donnée X :

$$\underbrace{p(\Theta|X,M)}_{\textit{Posterior}} = \frac{\overbrace{p(X|\Theta,M)}^{\textit{Likelihood}}\overbrace{p(\Theta|M)}^{\textit{Prieur}}}{\underbrace{p(X|M)}_{\textit{Evidence}}} \tag{25}$$

où la dépendance sur M est présentée explicitement. L'évidence est la probabilité d'observer les données d'un modèle particulier M. Bien que cette quantité ne soit pas importante dans ce premier niveau de l'inférence, elle joue un rôle crucial au deuxième niveau de l'inférence, qui est le choix de modèle. Pendant le choix du modèle, le postérieur du modèle ayant vu les données X est calculé par la règle de Bayes:

$$p(M|X) \propto p(X|M)p(M) \tag{26}$$

Dans la pratique, il n'y a aucune raison de favoriser un modèle à l'autre. Par conséquent, l'antérieur $p(M)$ est souvent choisi d'être uniforme, dans ce cas les modèles peuvent être rangés par leur évidence $p(X|M)$. Pour calculer l'évidence, cependant, nous devons intégrer sur les paramètres des modèles :

$$p(X|M) = \int p(X|\Theta, M)p(\Theta|M)d\Theta \tag{27}$$

Malheureusement, cette intégrale est généralement insoluble. Ensuite, cette issue est abordée au moyen d'inférence variationnelle. Les calculs dans le framework bayésien peuvent rarement être effectués exactement, en raison de la nécessité d'intégrer sur les modèles. L'approximation doit donc être faite. Les méthodes de Markov chain Monte Carlo et l'approximation de laplace sont deux solutions pour cette approximation. Le premier tente de réaliser les résultats exacts mais nécessite généralement de vastes ressources de calcul.

V.1.2 Méthodes variationnelles

Les méthodes variationnelles sont utilisées comme les méthodes d'approximation, dans une large variété d'arrangements. Le terme des méthodes variationnelles réfère à une grande collection de techniques d'optimisation. Dans chaque cas, l'application de méthodes variationnelles convertit un problème complexe en un plus simple, où ce dernier est généralement caractérisé par un découplage des degrés de liberté dans le problème original. Ce découplage est réalisé par une extension du problème à inclure des paramètres additionnels, connus sous le nom de paramètres variationnels, qui doivent être adaptés au problème actuel. La solution de problèmes variationnels est souvent exprimée en termes d'équations de point fixe qui capturent les conditions nécessaires pour l'optimalité (caractérisant localement les solutions optimales). Ce sont analogues à la mise le gradient à zéro pour optimisation ordinaire d'une fonction. Une méthode qui applique successivement différentes équations de point fixe fournit une manière commune de trouver des solutions aux problèmes variationnels à chaque fois qu'une solution de forme-fermée ne peut être trouvée.

Dans la dernière décennie, un certain nombre d'approches variationnelles ont été utilisés avec succès pour l'inférence et l'estimation dans les grandes models graphiques connectés aux probabilités de densité pour lesquels les calculs probabilistes exactes ne sont plus possible (par exemple [89]). Leur succès découle principalement de deux idées : d'abord, les problèmes d'inférence probabiliste se prêtent naturellement aux formulations variationnelles et, seconde, les problèmes d'optimisation variationnelles résultants, admettent les principes de la solution approximative. Les formulations variationelles en fait, facilitent naturellement trouver les solutions approximatives.

V.1.3 Inférence variationnelle ou Baysienne variationnelle

Les méthodes variationnelle peuvent être utilisées pour rapprocher les intégrales requises à l'apprentissage de Bayes [19, 8]. L'idée fondamentale est de rapprocher simultanément la distribution sur les deux états cachés et les paramètres avec une distribution plus simple, le plus souvent en supposant que les états cachés et les paramètres sont indépendants.

Ce framework facilite les calculs analytiques des distributions postérieures sur les variables, les paramètres et les structures cachés. Ils peuvent être employés à la limite inférieure de la vraisemblance marginale (c.à.d. "evidence") de plusieurs modèles en vue d'effectuer le choix du modèle, et souvent fournir une approximation analytique au postérieur de paramètre qui est utile pour la prévision. Les postérieurs sont obtenus par un processus itératif de type EM. C'est une alternative au méthode d'échantillonnage de Monte Carlo pour créer une distribution postérieure dont l'échantillonnage direct est difficile. Se concentrant sur le postérieur de paramètre, le résultant de son rapprochement est plus efficace que le laplace car la Hessian ne nécessite pas être calculée, et produit le postérieur non-triviales pour n'importe quelle taille de l'échantillon.

Dans l'inférence variationnelle, la distribution postérieure sur un ensemble de variables latentes $Z = \{z_1 \dots z_n\}$ et le paramètre Θ, pour certaines données X, est approchée par une distribution variationnelle:

$$p(Z, \Theta|X) \approx q(Z, \Theta). \tag{28}$$

La distribution variationnelle $q(Z, \Theta)$ est limitée à une famille de distributions de forme plus simple que $p(Z, \Theta|X)$. Ceci est fait en restreignant la gamme de fonctions dont l'optimisation est effectuée. Cette famille est sélectionnée avec l'intention que q peut être très similaire au postérieur vrai.

Nous pouvons décomposer la log-probabilité marginale en utilisant:

$$\ln p(X|M) = F_M(q(Z, \Theta)) + KL[q(Z, \Theta)\|p(Z, \Theta|X, M)] \tag{29}$$

où nous avons défini

$$F_M(q(Z, \Theta)) = \iint q(Z, \Theta) \ln\{\frac{p(X, Z, \Theta|M}{q(Z, \Theta)}\} \, dZ \, d\Theta \tag{30}$$

$$KL[q(Z, \Theta)\|p(Z, \Theta|X, M)] = -\iint q(Z, \Theta) \ln\{\frac{p(Z, \Theta|X, M}{q(Z, \Theta)}\} \, dZ \, d\Theta \tag{31}$$

Nous pouvons maximiser la limite inférieure $F_M(q(Z, \Theta))$ par optimisation par rapport à la distribution $q(Z, \Theta)$ qui est équivalente à minimiser la divergence de KL. Si nous permettons n'importe quel choix possible pour $q(Z, \Theta)$, le maximum de la limite inférieure se produit lorsque la divergence KL disparaît, qui se produit quand $q(Z, \Theta)$ est égal à la distribution postérieure $p(X|Z, \Theta, M)$. Cependant, nous supposons que le modèle est tel que le travail avec la vraie distribution postérieure n'est pas résoluble.

En variationnelle de Bayes (VB), une postérieur approximative est choisie de manière à ce que la limite inférieure est tractable par l'examen d'une famille restreinte de la distribution $q(Z, \Theta)$. Ensuite le membre de cette famille est obtenu pour que la divergence KL soit réduite au minimum. L'objectif est de limiter suffisamment la famille pour qu'elle comprenne que les distributions souples, tout en permettant la famille d' être suffisamment riche et souple afin de pouvoir fournir une bonne approximation de la vraie distribution postérieure.

Nous divisons les éléments de Z en groupes disjoints que nous dénotons par Z_i où $i = 1, \dots, N$. Nous supposons alors que la distribution variationnelle de postérieure q factorise par rapport à ces groupes, de sorte que :

$$q(Z, \Theta) = q_Z(Z) q_\Theta(\Theta) = (\prod_{i=1}^{N} q_{z_i}(Z_i)) q_\Theta(\Theta) \tag{32}$$

où la dernière équation est une conséquence des données X étant iid (indépendantes et identiquement distribuées). Ainsi, l'approximation variationnelle du postérieur commun suppose l'indépendance entre les paramètres et les variables latentes compte tenu les données observées. Autrement dit, le problème est converti en plus simple en découplant les degrés de liberté du problème original. Sous cette factorisation, l'approche bayésienne variationnelle maximise itérativement F comme une fonctionnelle de distributions libres, $q_Z(Z)$ et $q_\Theta(\Theta)$ conduit aux équations EM de bayésiennes variationnelles [13]:

$$\text{étape de VBE : } q_{z_n}(z_n) \propto exp(E_\Theta\{\ln p(x_n, z_n | \Theta, M)\}), \forall n. \tag{33}$$

$$\text{étape de VBM : } q_\Theta(\Theta) \propto p(\Theta|M) exp(E_Z\{\ln L_c(\Theta|X, Z, M)\}). \tag{34}$$

Dans ces équations, $E_\Theta\{.\}$ et $E_Z\{.\}$ dénotent respectivement l'espérance par rapport à $q_\Theta(\Theta)$ et $q_Z(Z)$. Dans les deux étapes VBE et VBM, la limite inférieure sur le log-évidence est maximisée en minimisant la divergence de KL entre la postérieure variationnelle factorisée et le postérieur (vrai) conjoint des variables latentes et les paramètres, ayant X :

$$F_M(q_{z_1}(z_1), \dots, q_{z_N}(z_n), q_\Theta(\Theta)) \tag{35}$$

$$= \ln p(X|M) - KL[q_Z(Z) q_\Theta(\Theta) \| p(Z, \Theta | X, M)]. \tag{36}$$

La postérieure factorisée est optimisée en termes de divergence de KL, de telle façon à avoir une bonne approximation de postérieure vrai, réalisant la limite aussi forte que possible. Le framework de VB nécessite de choisir une première distribution antérieure sur les paramètres. En appliquant l'étape VBE et l'étape VBM à plusieurs reprises, le postérieur variationnel est calculé. Dans la pratique, il convient de choisir les priors conjugués à la famille exponentielle. L'antérieur $p(\Theta)$, est dit conjugué à $r(x|\Theta)$ si le postérieur $q(\Theta|x) \propto r(x|\Theta)p(\Theta)$ est de la même forme que $p(\Theta)$. Donc l'apprentissage dans le cadre VB consiste simplement à mettre à jour les paramètres de prior aux paramètres du postérieur.

V.1.4 Mélange de gaussien variationnel

Pour appliquer la méthode de bayes variationnel au GMM, on peut regarder le GMM comme un modèle variable de latent dans le sens que nous ne savons pas qu'un point de données est généré par quelle composante. Pour chaque observation x_n nous avons une variable latente correspondante z_n comprenante 1-de-K vecteur binaire avec les éléments z_{nk} pour $k = 1, \dots, K$. Comme auparavant, on note l'ensemble de données observées par $X = \{x_1, \dots, x_N\}$ et on indique le variables latentes par $Z = \{z_1, \dots, z_N\}$.

En apprentissage de VB, le postérieur variationnel rapproche le postérieur de commun en supposant que les variables latentes et les paramètres sont indépendants :

$$q(Z, \Theta_N) = q_Z(Z) q_{\Theta_N}(\Theta_N). \tag{37}$$

Compte tenu de cette factorisation, la limite inférieure sur le log-évidence est soluble et l'espace est minimisé en minimisant la divergence de KL entre le postérieur vrai et le variationnel. Cela se fait par itération au moyen de l'algorithme VBEM:

$$\text{étape de VBE : } q_{z_n}(z_n) \propto \exp(E_{\Theta_N}\{\ln p(x_n, z_n|\Theta_N, M)\}), \forall n. \tag{38}$$

$$\text{étape de VBM : } q_{\Theta_N}(\Theta_N) \propto p(\Theta|M) \exp(E_Z\{\ln L_c(\Theta_N|X, Z, M)\}). \tag{39}$$

Dans ces équations, $E_Z\{.\}$ et $E_{\Theta_N}\{.\}$ sont respectivement l'espérance de $q_Z(Z)$ et $q_{\Theta_N}(\Theta_N)$. Remarquons que la postérieure $q_Z(Z)$ factorise (car X sont i.i.d.). En raison de cette forme factorisée, l'étape VBE pour la GMM se simplifie à

$$\rho_{nk} = q_{z_{nk}}(z_{nk} = 1) \propto exp(E_{\Theta_N}\{\ln \pi_k + \ln N(x_n|\mu_k, \Lambda_k)\}). \tag{40}$$

Afin de calculer l'étape VBE, nous devons connaître $q_{\Theta_N}(\Theta_N)$. En regardant l'étape de VBM, on peut voir que c'est particulièrement intéressante de prendre l'antérieur $p(\Theta_N|M)$ sur les paramètres comme étant conjugués à la famille exponentielle. Dans ce cas, les postérieurs et les antérieurs ont la même forme fonctionnelle. En conséquence, l'étape de VBM consiste à simplement mettre à jour les hyperparamèters d'antérieur aux paramètres du postérieur.

Le prieur de commun conjugué pour le GMM est le produit d'un prieur de Dirichlet commun sur les proportions du mélange et les distributions Gaussiennes-Wishart sur les moyens et les précisions de chaque composante gaussienne :

$$p(\Theta_N|M) = Dir(\pi|\alpha_0) \prod_{k=1}^{K} NW(\mu_k, \Lambda_k|\Theta_{NW_0}) \tag{41}$$

où $\Theta_{NW_0} = (\beta_0, m_0, \nu_0, W_0^{-1})$ sont les valeurs particulières pour les hyperparamèters. Comme l'antérieur est un antérieur conjugué, le postérieur commun a la même forme fonctionnelle et est donc également le produit d'un Dirichlet et les distributions Gaussiennes-Wishart :

$$q_{\Theta_N}(\Theta_N) = Dir(\pi|\alpha) \prod_{k=1}^{K} NW(\mu_k, \Lambda_k|\Theta_{NW_k}) \tag{42}$$

où $\Theta_{NW_k} = (\beta_k, m_k, \nu_k, W_k^{-1})$. À ce point, l'espérance dans l'étape de VBE peut être calculée car la forme du postérieur est connue. Alors après quelques algèbres nous pouvons identifier a partir de 39, les règles pour la mise à jour de VBM pour les hyperparamèters de l'équation 42.

V.1.5 Nombre de composantes dans le modèle de mélange de Bayes variationnel

Les méthodes habituelles pour la détermination de nombre de composantes (K) exigent qu'une gamme de modèles de différentes valeurs K soit formée et comparée. Le mélange bayésien variationnel, traite le coefficient de mélange π comme les paramètres et fait une estimation de leurs valeurs en maximisant la limite inférieure par rapport à π.

Après la convergence, il n'y a que le nombre de composantes pour lequel les valeurs de l'espérance des coefficients de mélange sont numériquement distinguables de leurs valeurs antérieures.

Les composantes qui ne fournissent pas une contribution poure décrire les données et ne jouent aucun rôle dans le modèle auront leur coefficient de mélange conduit à zéro pendant l'optimisation, et elles sont effectivement enlevées du modèle par la détermination automatique. Cet effet peut être compris quantitativement en termes de trade-off automatique dans un modèle bayésien entre l'ajustage des données et la complexité du modèle, dans lequel la pénalité de complexité se pose à partir de composantes dont les paramètres sont poussés loin des valeurs prieures.

V.2 Agrégation des mélanges

Maintenant, nous présentons une solution pour notre problème en utilisant la modélisation bayésienne du problème de groupement de composante de GMM et nous résolvons en utilisant une technique variationnelle de Bayes, appliquée au niveau des composantes.

Comme dans la section III.2.1, nous considerons deux nœuds, chacun porte les différents modèles probabilistes de mélange de gaussien, noté $M_1(x)$ et $M_2(x)$, associée à la même classe multimédia et donc la densité caché $p(x)$. Les mélanges peuvent être exprimés comme $M_k(x) = \sum_{i=1}^{m_k} w_k^i N_k^i(x), \quad k = 1, 2$ où $N_k^i(x)$ est une composante gaussienne dont la moyenne est μ_k^i et la covariance Σ_k^i et les w_k^i sont les poids scalaires. Le modèle M_k est estimé sur un ensemble de données de la taille, n_k situé sur le nœud k. $p(x)$ peut être estimé en concaténant les mélanges entrants:

$$M_c(x) = \frac{1}{n_1 + n_2}(n_1 \sum_{i=1}^{m_1} w_1^i N_1^i(x) + n_2 \sum_{i=1}^{m_2} w_2^i N_2^i(x)) \tag{43}$$

M_c est paramétrisé par $\Theta^c = \{\pi^c, \mu^c, \Sigma^c\ \}$. Ayant M_c, nous souhaitons déduire le mélange réduit M_r, régi par $\Theta^r = \{\pi^r, \mu^r, \Sigma^r\}$. Le nombre de composantes m^r dans M_r, devrait également être déterminé.

Une direction possible d'atteindre le mélange du réduction, consiste à minimiser les divergences de Kullback $KL(M_c \parallel M_r)$. Toutefois, l'absence d'une forme fermée en cas de divergence entre les deux modèles de mélange implique le contournement par les approximations (par exemple [70, 186]). De plus, puisque la divergence de KL est directement liée à la log-vraisemblance, la question essentielle de détermination de la complexité de modèle approprié (c.-à-d. m^r) est laissée ouverte.

L'approche que nous proposons, consiste à effectuer modélisation et l'évaluation bayésienne du problème de groupement de composante:

- la modélisation bayésienne incorpore automatiquement un critère pour évaluer correctement la complexité de modèle approprié,
- l'inférence approximative par l'approximation variationnelle fournit les estimations fiables pour Θ^r à un coût de calcul similaire à celui d'un algorithme EM. De plus, le groupement de composante et l'annulation explicite des composantes inutiles est effectué dans un seul processus, c'est-à-dire le balayage de toutes les complexités possibles de modèle est évité.

Nous montrons dans la section V.3 les problèmes d'estimation du mélange réduit en utilisant l'échantillonnage virtuel, puis dans les sections V.4 et V.5 nous employons le bayésien variationnel de GMM pour trouver le mélange réduit à partir de M_c.

V.3 Réduction de GMM vs. l'estimation de GMM à partir des données

Dans la section III nous avons parlé de l'idée de concevoir un algorithme, dans le modèle objectif M_r, qui doit être estimé à partir du modèle M_c plutôt que les données. Ici, de façon générale, l'idée est de concevoir un algorithm de type EM comme discuté dans [186]. Une originalité de notre arrangement et la solution, résulte de la nécessité d'estimer les paramètres du modèle réduit d'un modèle d'overcomplete, plutôt que les données, le dernier cas étant arrangement commun pour le travail existant dans GMM-Bayes variationnel. Néanmoins, notre problème peut être remanié dans les arrangements classiques de GMM-de-données comme suit : supposons que les données X ont été générées à partir de M_c , l'objectif est de trouver M_r qui maximise l'évidence $\ln p(X)$, sous la contrainte que les composantes de M_r, sont des combinaisons linéaires des composantes de M_c. Un peu de statistiques et de l'algèbre montrent qu'une estimation de vraisemblance maximale de M_r à partir de X, peut également être exprimée à partir des paramètres de M_c (eqs.(44) à (47)) et par conséquent aucun échantillonnage réel de M_c ne doit être fait.

$$z_{ij} = \frac{[N(\mu_i^c|\mu_j^r, \Sigma_j^r)e^{1/2tr(\Sigma_i^c\Sigma_j^r)}]^{\pi_i^c}\pi_j^r}{\sum_k[N(\mu_i^c|\mu_k^r, \Sigma_k^r)e^{1/2tr(\Sigma_i^c\Sigma_k^r)}]^{\pi_i^c}\pi_k^r} \tag{44}$$

$$\pi_j^r = \frac{\sum_i z_{ij}}{m^c} \tag{45}$$

$$\mu_j^r = \frac{\sum_i z_{ij}\pi_i^c\mu_i^c}{\sum_i z_{ij}\pi_i^c} \tag{46}$$

$$\Sigma_j^r = \frac{\sum_i(z_{ij}\pi_i^c\Sigma_i^c + zij\pi_i^c(\mu_i^c - \mu_j^r)(\mu_i^c - \mu_j^r)^t)}{\sum_i z_{ij}\pi_i^c} \tag{47}$$

dans cette approche nous avons besoin du nombre de composantes pour trouver le M_r ce qui n'est pas défini automatiquement. Mais nous utiliserons les mécanismes de groupement dans cette méthode pour trouver la résolution variationnelle dans la section V.5.

V.4 Formulation bayésienne de réduction de GMM

En général dans les problèmes de modèle de mélange, nous ne pouvons pas déduire Θ^r directement à partir des les observations (ici, Θ^c) et l'introduction des variables latentes est nécessaire. Notons $Z = \{z_{ij}\}, 1 \leq i \leq m^c, 1 \leq j \leq m^r$ les variables aléatoires binaires indiquant si la composante i de M_c est assignée à la composante j dans le mélange réduit M_r.

Nous complétons la spécification du modèle en définissant une densité antérieure sur Θ^r, choisie comme prieurs conjugués les plus pratiques:

- $p(\pi^r) = Dirichlet(\pi^r|\alpha)$. La distribution de Dirichlet est un point clé pour l'arrangement que nous proposons, car sa définition implémente Occam's Razor, c'est-à-dire qu'elle pénalise un mélange inutilement complexe. En effet, le choix $\alpha < 1$ produit une densité dont la valeur augmente pendant que plus de variables en π^r se rapprochent à 0, c.-à-d. quand les composantes non essentielles disparaissent.
- $p(\mu^r, \Lambda^r) = p(\mu^r|\Lambda^r)p(\Lambda^r)$, une distribution Gaussienne-Wishart.

Le modèle probabiliste est complètement décrit par la densité commune $p(Z, \Theta^r, \Theta^c)$, et notre objectif est double:
(a) construire le postérieur $p(Z, \Theta^r|\Theta^c)$. Dans la pratique, Z_{ij} est entièrement décrit par son espérance.
(b) trouver la structure de modèle qui maximise l'évidence $p(\Theta^c)$.

V.5 Résolution variationnelle

Manipulation exacte $p(Z, \Theta^r|\Theta^c)$ est untractable, donc nous recourons à une solution approximative, par l'inférence variationnelle. Dans de nombreux circonstances, cette méthode approche à la qualité de résultats de Monte-Carlo tout en préservant le coût modéré d'approximation du postérieur (par exemple BIC). On peut facilement décomposer l'évidence $\ln p(X)$ en deux termes, par l'introduction d'une distribution intermédiaire q :

$$\ln p(X) = F(q(Z, \Theta^r)) + KL(q \parallel p(Z, \Theta^r|X)) \tag{48}$$

où

$$F(q(Z, \Theta^r)) = \int q(Z, \Theta^r) \ln \frac{p(X, Z, \Theta^r)}{q(Z, \Theta^r)} \, dZ \, d\Theta^r \tag{49}$$

Cette décomposition montre que :
- $q(Z, \Theta^r)$ est une densité approximative pour le postérieur $p(Z, \Theta^r|X)$;
- KL est toujours positifs, $F(q(Z, \Theta^r))$ est une approximation inférieure pour $\ln p(X)$.

En conséquence, la recherche de $q(Z, \Theta^r)$ pour assortir les deux objectifs suivants est en fait pour le but :
- trouver $F(q(Z, \Theta^r))$ qui rapproche mieux $\ln p(X)$;
- trouver $q(Z, \Theta^r)$ qui rapproche mieux $p(Z, \Theta^r|X)$.

Le point entier de la décomposition ci-dessus est que, si le choix pour $q(Z, \Theta^r)$ est suffisamment contraint, l'optimisation de $F(q(Z, \Theta^r))$ est resoluble, alors que l'optimisation de $\ln p(X)$ pas. Une contrainte effective consiste à supposer que $q(Z, \Theta^r)$ peut être décomposé comme le produit des facteurs indépendants, tel qu'il est exprimé dans 50.

$$p(Z, \Theta^r|\Theta^c) \approx q(Z, \pi^r, \mu^r, \Sigma^r) = q(Z)q(\pi^r) \prod_{k=1}^{m^r} q(\mu_k^r, \Sigma_k^r) \tag{50}$$

Optimisation de $F(q(Z))$ peut être conduit à l'optimisation de chaque variable aléatoire à son tour, tout en faisant la moyenne par rapport à d'autres variables aléatoires (c.-à-d. espérance, donnée une estimation actuelle de la distribution pour ces autres variables aléatoires). Dans le cas actuel, cela revient à l'itération entre les deux premières étapes suivantes:
- étape variationnelle de E: ayant un postérieur approximatif $q(\pi, \mu, \Sigma)$ (par ses statistiques suffisantes), nous rapprochons les postérieurs $q(Z)$ sur les variables Z, ce qui est en fait entièrement décrite par $< z_{ij} >$, où $< . >$ dénote l'espérance.
- étape variationnelle de M: ayant un postérieur approximatif $q(Z)$, on calcule un approximatif postérieur $q(\pi, \mu, \Sigma)$. Puisque nous employons les priors conjugués, cette étape correspond à la mise à jour des paramètres du modèle. Les analyses de traçabilité apparaissent en développant 49 et en considérant l'espérance par rapport à Z.

- réduire le nombre de composantes de M_r, si approprié : si l'un des composantes k dans M_r, a un faible prieur $\pi_k^r \approx 0$, elle est supprimée du mélange. Rappelons-nous que le prieur Dirichlet tend à favoriser telles configurations.

Les expressions et la dérivation détaillées pour ces mises à jour peuvent être trouvées dans V.1.3 dans le cas d'estimation à partir des données. Le remplacement des statistiques des données par des eqs. 44 à 47) fournit les étapes variationnelles réelles de E et M.

V.6 Validation

L'exemple de l'identification distribuée de locuteur est pris dans toute cette section, car c'est une forme représentative où les mélanges gaussiens sont très populaires. Cependant la technique s'applique directement à un éventail de classes audiovisuelles. Pratiquement, l'algorithme EM avec de multiples initialisations est employé dans nos expériences afin d'alléger le fardeau des minimum locaux, mais d'autres techniques peuvent être employées. Nous avons évalué la capacité de chaque mélange sur le réseau pour le modèle de données D de la classe d'intérêt (ici, un locuteur) de $p(D)$, où D ici est l'union des données expédiées sur tous les nœuds, qui ne sont jamais recueillies quand le système pratique fonctionne, mais est un facteur de mérite pour l'observation externe. Nous avons mis en place une expérience impliquant sur 20 nœuds, sur Matlab. Le processus se stabilise autour d'un "modèle collectif" avec une très petite variation.

VI Conclusion

Dans cette section, nous résumons nos contributions principales.

VI.1 Contributions principales

Cette thèse aborde le problème de concevoir l'apprentissage distribué pour le système de reconnaissance de façon décentralisée. Dans ce travail, la répartition de calcul et de données est dû au contexte applicatif, dans lequel les systèmes indépendants coopèrent. Les algorithmes et des données de divers contributeurs coopèrent vers un apprentissage statistique collectif. Le but principal du système est d'obtenir une estimation dont la qualité est proche du modèle qui aurait été estimé dans une version centralisée. Malgré sa simplicité, cette technique asynchrone et décentralisée est très efficace. L'issue technique abordée par cette thèse est le calcul distribué d'une densité de probabilité.

Ses bonnes propriétés peuvent être résumées pour notre problème comme suit :

- accélération en implémentant le parallélisme de brut-grain (coarse-grain) sur l'ensemble de nœuds. Ceci se produit à deux niveaux :
  1. parallélisme de l'apprentissage basé sur la méthode gossip par la fusion
  2. pour chaque étape de l'apprentissage basé sur la méthode gossip, le parallélisme dans le calcul des probabilités pour la validation
- robustesse dans le calcul de distribution et d'estimation statistique, car:
  – n'importe quel nœud peut partir pendant le gossip sans causer la dégradation principale ou se joindre et obtenir, avec forte probabilité, une estimation efficace de ce qui a été précédemment estimé collectivement sur le réseau.

  - une très mauvaise estimation dans une minorité des nœuds n'affecte pas le modèle collectivement estimé.

L'efficacité de la technique proposée vient des deux caractéristiques principales suivantes:

- la fusion des estimations de densité entre les nœuds implique seulement la transmission de, et le calcul sur les paramètres de modèle de mélange, plutôt que généralement le grand nombre de données de multimédia (ou de vecteurs d'attribut qui le représentent). En conséquence :
  - la quantité de l'information à envoyer sur le réseau est très faible.
  - le calcul sur les nœuds reste relativement faible, par rapport aux tâches d'estimation qui opèrent sur les données de multimédia ou les vecteurs d'attribut.
- pendant la phase d'apprentissage de modèle basé sur la méthode de gossip, la complexité de tout mélange (c'est-à-dire le nombre de composantes gaussiennes) garde un ordre de grandeur. Soulignons que la phase d'apprentissage distribué et la phase de requête, peuvent entièrement recouvrir, car la réduction de mélange maintient la représentation de classe directement prête pour l'évaluation de requête.

Nos contributions principales sont les suivants :

• Tout d'abord, nous avons présenté une vue d'ensemble sur les techniques d'apprentissage automatique, l'indexation et la récupération de multimédia et les techniques de calcul distribué qui sont les éléments d'un apprentissage distribué pour les systèmes de reconnaissance multimédia. Un grand nombre de données étiquetées et non étiquetées de multimédia (dans le sens de l'apprentissage supervisé) est distribué sur Internet. La reconnaissance des structures de multimédia est généralement une tâche calculeuse et à usage intensif de données. La perspective d'exploiter collectivement ces données d'une façon décentralisée et distribuée est plus réaliste. L'indexation de multimédia est en grande partie une issue d'apprentissage/reconnaissance et nombreux logiciels et différents algorithmes de reconnaissance des structures ont été distribués sur l'Internet, mais ils fonctionnent en principe localement et de nombreux algorithmes de reconnaissance des structures sont exigeants en CPU et la structure de multimédia à indexer est énorme. Afin de permettre une évolution flexible de l'ensemble de classes, nous favorisons une approche générative, qui caractérise la classe dans l'espace d'attribut. Cette approche générative mène à des solutions plus souples, car l'introduction de nouvelles classes dans le système ne requiert aucune mise à jour à la description de classes connues. Pratiquement, nous estimons la densité de probabilité de la classe conditionnelle . Nous sommes concentrés sur le cas où toutes les densités sont les mélanges gaussiens, cette forme de modèle est omniprésent dans la modélisation des données multimédia, parce qu'il a de nombreuses bonnes propriétés. Nous avons également étudié à la calculation répartie en raison de leurs bonnes propriétés à s'occuper de grandes quantité de données d'apprentissage et de systèmes de reconnaissance distribués sur un ensemble de nœuds. Pour avoir les résultats généraux nous nous efforçons de développer les solutions algorithmiques communes avec le bon niveau d'abstraction du contexte. Ainsi, nous avons supposé un système de l'architecture distribuée pair à pair (P2P) qui est extensible à très grandes échelles.

Alors nous avons présenté "l'estimation distribué de mélange gaussien" comme un "apprentissage distribué" qui apprend un modèle en utilisant les données ou les modèles distribués sur le réseau sans rassembler l'ensemble des données sur un nœud central. Notre objectif est d'avoir un algorithme entièrement décentralisé qui calcule et dissémine les agrégats des modèles et de trouver un modèle global, avec un minimum de traitement et d'exigences de communication et du bon comportement de tolérance aux pannes.

• Deuxièmement, nous abordons le problème de l'estimation collective des paramètres et la complexité d'un mélange gaussien pour modeler la densité de probabilité de la classe conditionnelle. Nous avons proposé une solution pour l'apprentissage distribué en agrégeant les modèles de mélange qui ne requiert qu'un calcul modéré à chaque nœud et peu de données de transit entre les nœuds. Les modèles peuvent être estimés localement, par une technique classique. Nous nous sommes concentrés sur l'agrégation des modèles locaux pour améliorer la caractérisation de la classe. La caractéristique principale de notre approche est que les nœuds participants transmettent seulement les paramètres de modèles de mélange plutôt que des vecteurs d'attribut de multimédia, puisque l'agrégation exige seulement les paramètres modèles. En outre, une estimation du nombre approprié de composantes est effectuée régulièrement, afin de permettre l'éxtensibilité à grande échelle. L'arrangement proposé agrège le modèle de mélange local estimé en déterminant une combinaison appropriée de leurs composantes. Le mécanisme pour propager les mélanges entre les nœuds qui participent à l'arrangement, est gossip. N'importe quel nœud peut alors fournir, à tout moment, une estimation du modèle qui s'améliore avec le temps.

• Troisièmement, nous avons abordé le problème de trouver le meilleur modèle dans la méthode d'agrégation et nous employons la méthode de validation-croisée pour améliorer l'algorithme. L'agrégation de mélange ne garantit pas une meilleure vraisemblance du modèle de mélange résultant aux données entières comparant avec les deux modèles initiaux. Pour améliorer l'algorithme, nous employons une combinaison de la de validation croisée et la fusion afin de trouver le meilleur modèle parmi de nombreux modèles qui existent sur le réseau pour une classe unique. L'algorithmes utilise d'abord l'approche de validation croisée et trouve un modèle localement le meilleur dans une région selon les nœuds dans cette région (voisinages d'un nœud). Après cela, il essaie de trouver les meilleurs modèles par la fusion des modèles dans cette région. Ce modèle peut propager dans le réseau et concurrencer d'autres modèles localement meilleurs des autres régions dans le réseau.

• Quatrièmement, nous avons proposé une solution efficace pour agréger les modèles de mélange en utilisant la méthode du bayésien variationnel. Les méthodes variationnelles peuvent être utilisées pour rapprocher les intégrales nécessaires à l'apprentissage de Bayes. L'idée fondamentale est de rapprocher simultanément la distribution sur les deux états cachés et les paramètres avec une distribution plus simple, le plus souvent en supposant les états cachés et les paramètres être indépendants. Dans cette solution, l'agrégation est réalisée par la modélisation bayésienne du problème de groupement de composante de GMM et elle est résolue en utilisant une technique variationnelle de Bayes, appliquée au niveau de la composante. Ceci détermine, par un processus simple, peu coûteux et pourtant précis, les attributions des composantes qui devraient être agrégées et le nombre de composantes dans le mélange après l'agrégation.

CHAPTER 1

Introduction

1.1 Motivation

Multimedia retrieval is growing very fast and becoming an important task in modern application domains (including medical imaging, e-commerce, illegal copy search for copyright reasons, etc.) and general purpose applications (e.g. Goggle Image and Youtube). The term of multimedia includes a combination of texts, images, audio and video documents. A central and classical need expressed by content-based indexing of multimedia documents is the assignment of a symbolic class label to a document or portion thereof, such as identifying a face, a speaker or a spatio-temporal texture or event [56]. Building a search engine able to recognize very many kinds of such audiovisual classes is a formidable task, long rated as unrealistic by the computer vision community, that is currently reviving as one of the most stimulating visions for both research and applications in the field [144].

Characterizing classes is the job of classifiers. The general task for training these classifiers is supervised learning that needs labeled data for training. This requires a careful design of media-specific observations from the raw data. Usually, more features are required as there are more classes to be distinguished which in turn increases the amount for training data needed and the computation power required. There are, however, encouraging trends towards the perspective of automatic large-scale harvesting of training examples : (i) joint text/image analysis, which may be fed by a massive resource of web pages or video OCR [162], (ii) recent advances in weakly supervised learning [163, 196] which enables learning a class from instances supplied in clutter (e.g. a face within a complex background). This suggests that the (necessary huge) amount of training data would inherently be distributed on a large scale and provided by independent sources. This large amount of data can be used to learning in semi-supervised manner [42], which can handle jointly class labeled and unlabeled data in the training phase.

On the other hand, end-hosts are becoming more powerful and provide more services and services on the internet are evolving from centralized client-server architectures to fully distributed architectures. In last decade, many distributed systems have been deployed in a wide variety of environments, e.g. peer-to-peer (P2P) networks, computing grids, sensor networks, etc. In a P2P organization [123] (i) resources are dynamic: data and learning/classification services can join or leave the network at any time; (ii) a node is both client and server : it can produce some services and use the services on the other nodes ; (iii) resources are aggregated: the quality of the global service is due to its collective aspect; (iv) the system is decentralized: each contributor can supply data or learning tools, without any central administration.

Thus, we have such large-scale and decentralized networks in which we can find, on one side a large amounts of labeled and unlabeled multimedia data (in the sense of supervised learning) and on the other side many softwares and different pattern recognition algorithms (which are trained mainly using local data and work locally). The two elements, multimedia data and pattern recognition algorithms are the two main elements for making network-global learning, but it lacks coordination. Work at the intersection of these fields includes, for instance, efficiency and data placement related to search of similar images [126] and collective learning from text data [175]. We can not transfer huge data of multimedia on the network to accumulate them on a central node for training a classifier. This motivate designing a distributed learning system to learn from whole data without transferring data on the network.

1.2 Thesis objective

In this thesis, we consider that machine learning tools are made available online by various human contributors on the nodes as services. We consider that elements of interest are classes, i.e. exhibit a variability that should be modeled. Multimedia data is also placed on nodes and part of it is assumed to be labeled with its class, i.e. can contribute to supervised learning. We also assume the feature space to be common to all nodes in the network. Each of these learning services use local multimedia data to train its classifier but they would like to cooperate to improve their classifier. This can be seen that the classifiers would learn from each other. In other words, in a decentralized fashion, algorithms and data from various contributors would cooperate towards a collective learning. We focus on statistical supervised learning, although unsupervised classification is equally interesting. Each service can receive multimedia data from a user or other nodes and then send the corresponding class identifier (e.g. name of speaker), and/or request, or share training data for supervised learning. Our goal is to argue distributed learning system on the network which means to profit whole data on the network for learning. Both for alleviating the computational cost of learning and for reducing the amount of data on the network, we examine the case where supervised learning itself is distributed and, more precisely, decentralized. In a decentralized fashion, algorithms and data from various contributors would cooperate towards a collective statistical learning. One way to perform this cooperation is using the parametric models of the same class that are achieved from different sources and try to merge these models. Similar ideas are also being examined for collective learning from text data [175] and sensor networks ,[87, 135].

A suitable organization for the above vision is a peer-to-peer architecture which nodes would run a service providing supervised learning of a multimedia class and would possibly store some training data. Upon request, it could classify incoming data to the best of its current knowledge. The nodes can learn from each other by sharing their knowledge. We may have a better learning system if the knowledge of the peers combined with each other. As this is a broad perspective, we now restrict the thesis to decentralized supervised learning of a class. We do not address here in important issues such as service and data localization, elaborated placement schemes (examined in [126] for retrieval of similar images), the fact that class identifiers should conform to a standard, nor the query phase.

The set of classes may not known a priori in our scenario (e.g. new speakers may be introduced). To allow for a flexible evolution of the set of classes, we favor a generative approach, that characterizes the class in feature space, over a discriminant approach, that learns directly to distinguish it from other classes. This generative approach leads to more tractable solutions, as introduction of new classes in to the system does not require any update to description of known classes. This simplifies the system, which may then be viewed as many independent concurrent subsystems as there are classes. Consequently, in this thesis we will describe the techniques for a single class. As this is a broad perspective, we restrict to decentralized learning of a class. Practically, we estimate the class-conditional probability density. More precisely, we focus on learning a probability density in a feature space. This space is assumed common to all contributors in the network, for the class to be learned. There are in fact cases where this strong hypothesis is reasonable, such as the widely used mel-cepstral acoustic features for the speaker recognition task, which we apply our proposal to. We address the distributed computation of a probability density with the applicative motivation of multimedia document retrieval and in the particular context of a decentralized, distributed system.

Gaussian mixture model (GMM) is a semi-parametric form and good choice for modeling multimedia data as it has numerous good properties for modeling multimedia, thus, we propose the techniques to combining the models in this important case. Models can be estimated locally, by a classical technique for example EM algorithms. We aim to find a collective model from the individual models which is better than all models exist on the network. To achieve our goal, we examine the aggregating various Gaussian mixture models related to a class which are distributed on the network. The proposal implies sending only a small amount of information on the network and low computation cost on nodes, leading to an overall very fast scheme. In fact, sharing of class representation is carried out through model parameters, while we do not require transmission of, nor computation on, the generally large amount of multimedia data (or feature vectors that represent it).

In this system we employ a simple gossip (or epidemic) spreading procedure to aggregate the models. In these protocols, in each round, each node contacts one or a few nodes (generally chosen at random), and they exchange information. It enables fast and robust propagation in an asynchronous, decentralized manner that fits well the peer-to-peer viewpoint. High fault-tolerance and self-stabilization are the results of dynamics of information spread in gossip [17, 46, 181]. They are simple to implement, have moderate overhead , can scale to a huge number of nodes while usually do not need error recovery mechanisms and have high stability under stress and disruptions. We may summed up the properties of using gossip for our problem as follows :

- speed up by implementing coarse-grain parallelism over the set of nodes.
- robustness both in the distributing computing and statistical estimation senses, since :
  - any node may leave during the gossiping without causing major degradation or join and obtain, with high probability, an effective estimate of what has been previously collectively estimated on the network,
  - a very poor estimate in a minority of nodes does not affect the collectively estimated model.

In this work, the distribution of computation and data is due to the applicative context, in which independent systems cooperate. The key goal of the system is to obtain an estimate which quality is close to what would have been estimated in a centralized version. Despite its simplicity, this asynchronous, decentralized technique is very effective. Let us underline that the distributed learning phase, which we address in this work, and the querying phase, can fully overlap, as mixture reduction keeps the class representation directly ready for query evaluation. Finally, let us note that transmission of the data could be avoided in a much simpler way, by transmitting model parameters and sampling from them, but this implies considerable computation in large dimension spaces. In next section we describe the contributions of this thesis.

1.3 Chapters outline

This work has been done in the context of distributed learning for multimedia classes. The main objective is to find an approach to aggregate and manage the probabilistic models which are distributed on the network in order to provide more accurate and robust multimedia recognition system. The thesis is structured into four main chapters (Chapters 2 to 5) that are followed by a short conclusion (Chapter 6) where the main findings and achievements of the research are summarized and suggestions for possible further work are given.

• In chapter 2, **”Background”**, we first present an overview on the elements of a distributed learning for multimedia recognition systems which are machine learning techniques, multimedia indexing and retrieval techniques and distributed computing then we present the ”distributed estimation of gaussian mixtures” which tries to estimates gaussian models in a distributed data environment.

In this chapter we start our discussion by the techniques of machine learning and pattern recognition because multimedia indexing is largely a learning/recognition issue. A major need of multimedia indexing and retrieval is related to characterization of classes (defining observations, capturing variability, etc.). To allow a flexible evolution of the set of classes, we chose Gaussian mixture model. This model form has many good properties and has widely been used to model many multimedia classes [56, 80, 148]. Gaussian mixture models is a generative approach, that characterizes the class in feature space and learns directly to distinguish it from other classes. This generative approach leads to more tractable solutions, as introduction of new classes in to the system does not require any update to description of known classes (unlike the discriminant approaches). In this chapter we present the properties of Gaussian mixture models and the approaches for learning its parameters. Consequently, the remainder of this thesis, we focus on the case where all densities are Gaussian mixture models.

Next, we present in this chapter the general techniques of content-based multimedia indexing and retrieval. Media analysis and retrieval systems have received widespread public and media interest of late. There have been extensive studies on the design of automatic content-based indexing and retrieval systems. These methods focus on searching the contents, mainly based on the features of media (for example color, shape, texture and motion in visual media and phonemes, pitch, rhythm and cepstral coefficients for audio/speech) but in this thesis we focus on searching by classes. In fact we interest in different descriptions of a unique class to find a better description of it.

Multimedia pattern recognition is generally a computation and data-intensive task. The perspective of exploiting collectively this data in a decentralized, distributed manner is made more realistic, for instance by current research efforts (e.g. in [163]) towards the ability to learn class models despite clutter, thereby easing database collection. In chapter 2 we also take a look in distributed computing because of their good properties to deal with large amount of data learning and recognition systems distributed on a set of nodes. There are different distributed system contexts where we can use in our problem. However, to yield general results, we strive to develop common algorithmic solutions with the right level of abstraction from the context. Thus, we assume a peer-to-peer distributed system architecture which is able to scale up to very large configurations. Gossip-based protocols which use decentralized algorithms are good choices for information exchanging and some computing processes such as aggregation on distributed environments. We present the gossip algorithms because of their properties.

As a result, large amounts of labeled and unlabeled multimedia data (in the sense of supervised learning) is distributed on the Internet in a large-scale, decentralized manner. Many softwares and different pattern recognition algorithms exist on the Internet, but they are trained mainly using local data. Numerous pattern recognition algorithms are CPU demanding and the multimedia pattern to index is huge. Thus, we need a decentralized learning systems for multimedia data which can use the data distributed on the network for training a model for a class of data. At the end of chapter 2, we present "distributed estimation of Gaussian mixtures" which use the gossip algorithm to find a Gaussian mixture model in a decentralized manner and in a distributed environment. In this decentralized learning method, the nodes participate to find a Gaussian mixture model without collecting the whole data on a central node.

• In chapter 3, **"Low-cost distributed learning of a probabilistic mixture model"**, we will tackle to find the techniques for the problem of estimating the parameters of a GMM for a collective learning. We propose a solution for distributed learning by aggregating the mixture models which only requires moderate computation at each node and little data to transit between nodes. We consider a set of nodes and a certain amount of labeled training data on each node. Models can be estimated locally, by a classical technique. So far, our focus has not been on locating suitable nodes in a network, but on aggregating local models to improve class characterization. The main feature of our approach is that, in order to keep network and computation load moderate, participant nodes only transmit mixture model parameters rather than multimedia feature vectors, since aggregation only requires model parameters. Besides, an estimation of the appropriate number of components is carried out regularly, to enable scaling up. The proposed scheme aggregates local mixture model estimates by determining a suitable combination of their components. In fact, this combination is obtained by optimizing a modified Kullback divergence between the aggregated model and the concatenation of local models, through an iterative scheme. The mechanism for propagating mixtures between cooperating nodes that participate in the scheme is gossip. Gossiping, here, is a non-ending background process in which acquainted nodes may share their models. Any node may then supply, at any time, an estimate of the model which improves over time.

• In chapter 4, **"Choosing the best models by cross-validation method"**, we address the problem of finding the best model in aggregation method and use the cross-validation method to improve the algorithm. Aggregating two mixture models did not guarantee that the resulting mixture model has better likelihood to the whole data than the two models. The problem is that the system can not distinguish the good models from the bad models. In fact for some of the nodes, the model parameters may have been not estimated well and it can affect on the other mixtures when merging is occurred. For improving the algorithm we use a combination of cross validation and merging to find the best model among many models which exist on the network for a unique class. The algorithms first use cross-validation approach and find a locally best model in a region according to the nodes in that region (neighborhoods of a node). After that it tries to find better models by merging the in that region models. This model can propagate to the network and compete to the other locally best models from other regions in the network.

• In chapter 5, **"Decentralized learning of a Gaussian mixture using variational Bayes-based aggregation"**, we propose an effective solution for aggregating the mixture models using variational Bayesian estimation. Variational methods can be used for approximate the integrals required for Bayesian learning. The basic idea is to simultaneously approximate the distribution over both hidden states and parameters with a simpler distribution, usually by assuming the hidden states and parameters are independent. In this solution aggregation is achieved through Bayesian modeling of the GMM component grouping problem and solved using a variational Bayes technique, applied at component level. This determines, through a single, low-cost yet accurate process, assignments of components that should be aggregated and the number of components in the mixture after aggregation.

CHAPTER 2
Background

2.1 Introduction

In the past decade, we have had advances in the three major technologies : VLSI that produce greater processing power, broad-band networks that provide much higher bandwidth and multimedia compression techniques which enable efficient storage and communications. These advances caused the rapidly use of digital multimedia media which includes creation, processing and transmission of high-volume multimedia data over the networks. Meanwhile, these multimedia documents are more and more accessible to the users which grow quickly with the rapid development of the Internet. As the multimedia databases become more larger and widespread, we need multimedia retrieval systems to be more efficient. Most multimedia retrieval systems usually have the following two steps for searching in their database:

1. indexing: in which a feature vector from the essential properties is computed and stored for each multimedia document in the database ,
2. searching: given a query, first its feature vector is computed and after comparing to the feature vectors in the feature base, the multimedia document or documents which are most similar to the query are returned.

We will present the general techniques for multimedia content based retrieval in section 2.3

Most of indexing and searching systems for multimedia documents are based on statistical learning algorithms. To improve the multimedia indexing and retrieval systems, we need to find (i) better learning algorithm and (ii) more data to feed these learning algorithms. The learning algorithms concerned with the study of computer algorithms for improving automatically through experience. One of the powerful approaches in the machine learning is using Gaussian mixture models (GMM) for modeling the classes of multimedia. It is used in many multimedia modeling because of its good properties specially its accuracy in density modeling and good behavior in high dimension space. The machine learning and Gaussian mixture model will be presented in section 2.2

In the other side, nodes have more and more power and can contribute in distributed computing. The networks change rapidly from centralized client-server architectures to fully distributed architectures like peer-to-peer networks. Various types of distributed systems, services and applications have been developed and are being used extensively in the real world. We can find a huge multimedia data and learning algorithms on the nodes of such networks which can be used for a global learning. In section 2.4 we will take a look to distributed computing and peer-to-peer systems.

Finally, we can use these powerful nodes which are distributed on the network to improve the performance of learning for multimedia documents. One of the problems with increasing of distributed data is how to learn or find the models from these distributed data. In fact we can use this configuration to make a ”distributed learning” system. This distributed learning system can be defined as a multimedia learning system which estimates the models using the data distributed on the network without collecting the whole data on a central node. We like to have a fully decentralized algorithm that computes a model with minimal processing and communication requirements and good fault-tolerant behavior. When we have GMMs as learning systems on the nodes, the distributed learning can be reached from distributed estimation of these models. We will talk about distributed estimation of GMM in the section 2.5

2.2 Machine learning

2.2.1 Definition and types

One of the subfields of artificial intelligence is machine learning. It concerns with the study of computer algorithms for developing the techniques that allow computers to "learn" which means improving them automatically through experience. In practice, this can be done through creating programs that analyze the data and optimize a performance criterion. On general, we have two types of learning: inductive, and deductive. In deductive reasoning the previously facts is used for new conclusion. The logical statements are combined using certain rules in order to find new statements. This is the way to prove theorems from axioms.

Constructing the axioms from the observation is called inductive reasoning. In fact, the observations are supposed to be consequences of these axioms. This is the same way that the physics scientists do. They observe natural phenomena then they conclude the laws of nature. We can create computer programs, using inductive machine learning and by extracting rules and patterns out of massive data sets. Statistics have a direct role in many machine learning algorithms. The machine learning research is also focused on the computational properties of the statistical methods, like their computational complexity.

There are many applications which use machine learning. Search engines, medical diagnosis, speech and handwriting recognition, natural language processing, object recognition in computer vision and classifying DNA sequences are some of examples. Machine learning is a branch of theoretical computer science known as computational learning theory. Common algorithm types include:

supervised learning: in supervised algorithms, we have a number of the training pattern which their labels are determined and represents the category of the pattern. In other words, training data consist of pairs of input objects, and desired outputs. The algorithm generates a function which maps inputs to desired outputs. After training with a number of training examples, the function should have the power to predict the value for any valid input object. The learner should find the values for unseeded data from the presented data in a "reasonable" way. If the output of the function has a continuous value we call it regression and if it predict a class label of the input object it is called classification.

unsupervised learning: in unsupervised learning the labeled examples and sometimes the number of classes are not available. The algorithm is presented only with examples from the input space. A model then is fit to these observations [82]. A clustering algorithm is un example of unsupervised learning.

semi-supervised learning: in semi-supervised learning, for generating the appropriate function or classifier, both of labeled and unlabeled data are used for training (typically a small amount of labeled data with a large amount of unlabeled data) [20, 35]. Many machine-learning researchers have found that it can produce considerable improvement in learning accuracy if we use unlabeled data in conjunction with a small amount of labeled data.

reinforcement learning: in this type of learning, no desired signal is given. But the environment provides a feedback that guides the learning algorithm. In the other word, it receives a reward, which may be either positive or negative. The feedback says that the tentative category is right or wrong but it is not told how. Using this feedback, the algorithm learns how to act given an observation of the world [66, 92, 173]. The reinforcement learning is usually suited for the problems which include a long-term versus short-term reward tradeoff such as chess, elevator scheduling and robot control.

transduction: it is like a supervised approach but it does not explicitly construct a function. It is used in the cases where any inductive model can be produced. Instead, the output is predicted based on the training inputs, training outputs, and new inputs [182]. It can be viewed as reasoning from observed training data which makes general rules and then applied to the test cases.

meta learning (learning to learn): this type of algorithms are self-adaptive and can learn their own inductive based on previous experience. The goal of meta-learning is to build a learning system that adapt to its specific environment which may be dynamic [110]. The meta-learning mechanism must improve the performance of the system by updating the current learning strategy when the experiences or meta-knowledge are available.

2.2.2 Statistical estimation

When we have a collection of data, one way to describe or summarize them is the statistical methods. Using statistical methods, we can recognize the probabilistic nature both of the information we seek to process, and of the form which we should express the result. If we have a statistical model in hand, we can get an algorithm for it, using probability theory and decision theory. This is opposed to using merely the training data to design an algorithm by selecting it among different algorithms or using heuristics/”common sense”. Providing a framework for studying the problem of inference is the main goal of statistical learning theory. It contains gaining knowledge, making predictions, making decisions or constructing models from a set of data [182]. This theory offer an introduction to some of the theory and methodology of modern classification and statistical learning and the parameters in these approaches, can be estimated by a variety of methods.

In the statistical approaches, each pattern is represented as a point in a d-dimensional space or in terms of d features. It is important to choose those features that can distinct the pattern vectors belonging to the different categories. We can determine the effectiveness of the representation of the feature set by how well it separates the patterns from different classes. As we can see in figure 2.1 the recognition system in a statistical pattern recognition is operated in two modes: training (learning) and classification (testing). The preprocessing module attempts to segment the pattern of interest from the background, remove noise, normalize the pattern, and

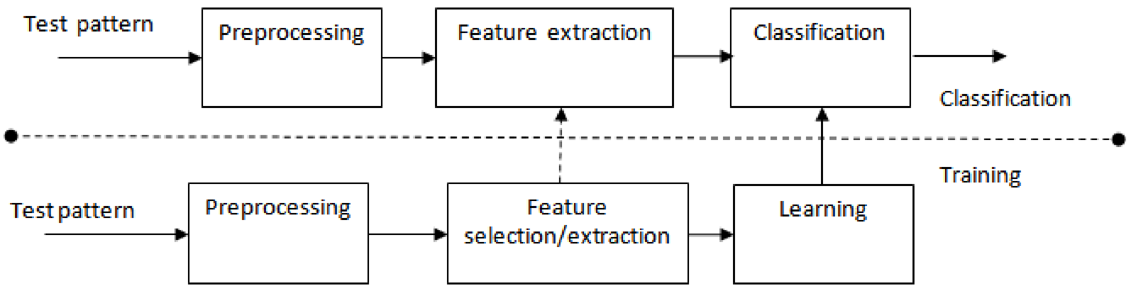


Figure 2.1 – Model for statistical pattern recognition

any other operation that can be helpful in defining a representation of the pattern. In the training mode, first the appropriate features will be find in the feature selection/extraction module to represent the input patterns. Then the classifier is trained by learning module to partition the feature space. In the classification mode, one of the pattern classes is assigned to the input pattern by the trained classifier based on the measured features.

Depending on the kind of information available about the class-conditional densities, various strategies are utilized to design a classifier in statistical pattern recognition. Some of important statistical learning systems are: linear classifier [18, 53], support vector machine [40], relevance vector machine [178], mixture models, mixture of experts [91] and decision tree [26].

2.2.2.1 ***Probability density estimation***

Consider a continuous random variable X that describes an unknown process. The density function or probability function or probability density function (PDF) is a usual means to specify this variable. The PDF provides a very rich source of information of underlying process. Besides, we can use the PDF to determine in which portion of space X can take a certain value and with which probability. In practice we have only a few noisy realizations of X and the true PDF is unknown. Therefore, we estimate an imperfect PDF based on the observations. For constructing an estimate of a PDF model, we use a density estimator to extract the information and key quantities embedded in raw data such as the mean, the most probable value (mode), the dispersion around the mean (variance), the degree of asymmetry (skewness) and many other characteristic quantities. The goal of density estimators is to determine the distribution of data within the input space or to estimate the probability density of given data. Thus, PDF of a continuous random variable is a probability distribution for the random variable which by integrating we obtain the probability of the random variable in a given interval and has the following properties:

$$(a) f(X) \geq 0 \quad \forall X \tag{2.1}$$

$$(b) \int_{-\infty}^{+\infty} f(X)\, dX = 1 \tag{2.2}$$

2.2.2.2 *Parametric versus nonparametric*

When performing density estimation, two major families of methods can be considered : the parametric and nonparametric ones. Another family between these two groups is called semi-parametric methods.

Parametric PDF estimation assumes the data is drawn from a specific density model and the unknown PDF is represented in terms of a priori chosen functional form which contains a number of adjustable parameters for example mean and variance in the case of Gaussian family. We need a procedure for determining suitable values for the parameters from observed data. To give the best fit to the data, the values of the parameters can be optimized. Maximum likelihood, maximum a posteriori, predictive estimation, cross validation, bootstrapping, Expectation-Maximization (EM) and boosting are the examples for optimizing parametric models. Maximum likelihood (ML), search for parameter values that maximize the likelihood function. ML ignores the prior distribution. Maximum a posteriori is another learning algorithm which seeks the parameter values that maximize the posterior density of the parameter. This implies that you have a prior distribution over the parameter. Monte Carlo integration is a numerical approach, which after sampling candidate models it take an average of their predictions. In the cross validation method, the training data is divided into several parts; in each turn, it use one part to test the procedure fitted to the remaining parts. It can be used for parameter estimation when there are many parameters.

One limitation of the parametric approaches is that they assume a specific functional form for the distribution. This may not be appropriate for a particular applications and might give a false representation of the underlying process. An alternative approach is nonparametric density estimation methods. In these model structures, a priori is not defined. The form of the distribution in these approaches typically depends on the size of the data set. In non-parametric methods there is no need to choose a model; we simply estimate the density at each point according to the input vectors in its neighborhood. Such models still contain parameters, but these parameters control the model complexity rather than the form of the distribution. Non-parametric methods can be applied to a much broader range of applications and are suitable to model any arbitrary density. Estimating these models usually needs a large amount of data. Histograms, nearest-neighbor density estimation and kernel density estimation are examples of non-parametric models. In nearest-neighbor density estimation, the density at any point is inversely proportional to the distance to the kth nearest datum. This method is usually combined with feature selection because it is very sensitive to irrelevant features [53]. In kernel density estimation, a kernel function is centering on each data point and sum of these functions give the density. The estimation quality and the performance depends on the kernel function and if the proper kernel is not chosen, it can not achieve to the excellent performance.

The parametric estimators are simple but not flexible, by contrast, the nonparametric estimators are very flexible but their statistical precision decrease greatly if we include several variables in the model. Consequently, researchers have tried to develop models and estimators which comprises the two aims, flexibility and simplicity of statistical procedure. This has led to a third family, called semiparametric, which is between parametric and nonparametric. Semi parametric estimators usually combine features of parametric and nonparametric techniques so they have parametric (finite dimensional) components and nonparametric (infinite dimensional) components. They overcome the problem of dimensionality by employing some form of dimension reduction.

2.2.2.3 ***Discriminative versus generative***

Within a statistical framework, the dependence of an unobserved variable y on an observed class variable x can be modeled by discriminative models. In the other words, discriminative approaches are used to model the prediction of the conditional probability distribution $P(y\|x)$. All the variables in this type of modeling are directly measurable. In the other side, generative models describe or estimate a probability density over the variables and can be used to manipulate nondeterministic models. They describe a model for randomly generating observed data when we are typically given some hidden parameters. These types of modeling, defines a joint probability distribution over data observed. It may be used for modeling data directly (to model data from a probability function) or as an intermediate step to create a conditional probability density function using the bays rule. If we have a system with variables $x_1, \dots, x_T$, it can be specified through a joint probability distribution over all of its variables $(p(x_1, \dots, x_T))$. Having knowledge about the problem at hand, the generative models give the user the ability to seed the learning algorithm for example in terms of structured models, independence graphs, Markov assumptions, prior distributions, latent variables, and probabilistic reasoning.

One of the important different between discriminative and generative models is in the optimization algorithms. Discriminative approaches adjust a possibly non-distributional model to data and optimizing for a specific task, such as classification or prediction. An example is support vector machines [40] which directly maximize the margin of a linear separator between two sets of points in a Euclidean space. Generative models uses the generic optimality criteria such as maximum likelihood. Other difference is that in generative models, it is possible to generate samples of various configurations of the system since we have the probability distribution but generating samples from the joint distribution in a discriminative model is not allowed. In summary we can say that the generative models try to describe a phenomenon and to generate configurations from it. They provides a rich framework for imposing structure and prior knowledge on a given problem and has more flexibility than discriminative approaches. If the model is not complex, the discriminative approaches is appropriate for it but in complex models the generative usually can better model data and optimize it.

2.2.3 Gaussian Mixture Model

The mixture of Gaussians is a generic and semi-parametric solution. In this model we use many local models to create the global model. It is a simple linear superposition of Gaussian components that aimed at providing a richer class of density models than single Gaussian. One of the important properties of this model is that it is ubiquitous in modeling of multimedia data, due to its numerous good properties (density modeling accuracy, good behavior in high dimension space, clean procedures for estimation and model complexity determination). They have widely been used to model e.g. audio classes [148], images [80] or motion-based spatio-temporal events in videos [56].

Next, we investigate this model in more detail.

2.2.3.1 ***The Gaussian distribution***

The Gaussian distribution, also known as normal distribution plays a central role in the statistical applications. It is widely used for modeling of distribution of continuous variables. In the case of single variable x, the Gaussian mixture distribution can be written in the form:

$$\mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{(2\pi)^{d/2}\sigma^{1/2}} \exp((-1/2\sigma^2)(x-\mu)^2) \tag{2.3}$$

where μ is the mean and σ^2 is the variance. If we have a d-dimension vector x, the multivariate Gaussian distribution takes the form:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \exp((-1/2)(x-\mu)^T \Sigma^{-1}(x-\mu)) \tag{2.4}$$

where here μ is a d-dimensional mean vector, Σ is a $d \times d$ covariance matrix, and $\mid \Sigma \mid$ denotes the determinant of Σ.

2.2.3.2 ***Mixture models***

One of the approaches for density estimation is mixture models, which is the task of finding a suitable approximation for the density. The simplest approach is to draw a hypercube around each observation. This results in an histogram of the data. Mixture modeling (or finite mixture modeling) is a weighted sum of other distributions which is used for modeling a statistical distribution by a mixture. By the mixture models we obtain a complex model which is based on several functions of densities of probabilities. In other words, the probability density expressed as a weighted sum of simpler densities. The simpler densities are called components or states of the mixture and by combining, we have a multimodal density.

If we have two random variables X and Z and the density of Z is $h(z)$ and the density of X conditionally has Z is $N(x|z)$, then the density of mixture is defined by:

$$f(x) = \int N(x|z)h(z)\,dz. \tag{2.5}$$

If Z is defined on a finished whole of values $1, \ldots, K$, distribution of Z is discrete and has a positive probability for each one of these values. So the density of the mixture is

$$f(x) = \sum_{k=1}^{K} p_k N_k(X) \tag{2.6}$$

with

$$N_k(x) = N(x|Z=k) \quad and \quad p_k = P(Z=k), \tag{2.7}$$

where the p_k is the proportions of mixture checking the following constraints: $0 \leq p_k \leq 1$ for any $k = 1, \ldots, K$ and $p_1 + p_2 + \ldots + p_K = 1$. If we use Gaussian components for modeling, it recognizes as mixture of Gaussians [19].

2.2.3.3 ***Gaussian Mixture Model (GMM)***

The Gaussian distribution has some important analytical properties and can be used in many contexts but when it comes to modeling real data sets it suffers from significant limitation. If the data sets form from two or more dominant clumps, a simple Gaussian distribution can not efficiently model this structure. But a linear superposition of some of Gaussian distributions gives better characterization of the data set. It means that if there are N input vectors in R^d, in order to estimate the input's probability density, a linear combination of K basis functions will be used as follows:

$$p(x) = \sum_{k=1}^{K} p(x|k)p(k) \tag{2.8}$$

where $p(x)$ is the density at point x. The k-th component density is shown by $p(x/k)$ and $p(k)$'s are the mixing coefficients, which means the prior probability of a data vector having been generated from component k of the mixture. The constraints for this model are:

- $0 \leq p(k) \leq 1$
- $\sum_{k=1}^{K} p(k) = 1$
- $\int p(x|k)\, dx = 1$

Main Properties:
Using a mixture model of almost any continuous density of component density function we can approximate a continuous density to arbitrary accuracy if:

1. K is sufficiently large,
2. The parameters of the model are chosen correctly.

It is said a Gaussian mixture where we have a linear and sufficient number of combination of Gaussians and adjusting their means, covariances and the coefficients. Gaussian Mixture Model is a distribution which probability density function of kth component (as well as its likelihood)is a Gaussian distribution:

$$p(x|k) = \frac{1}{((2\pi\sigma_k^2)^{d/2})} \exp \frac{\| (x-\mu_k)^2 \|}{2\sigma_k^2}. \tag{2.9}$$

So the mixture of Gaussians is defined as a superposition of K Gaussian densities of the form

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k). \tag{2.10}$$

Each Gaussian density $\mathcal{N}(x \mid \mu_k, \Sigma_k)$ is a component of the mixture and has its own mean μ_k and covariance Σ_k. The parameters π_k are called mixing coefficient and are the weight or density of components.

2.2.3.4 ***Parameters of the Gaussian models in the GMM***

Gaussian models correspond to a whole of constraints on the matrices of variance and the proportions of mixture. A component k of the model is defined by:

- **center** μ_k: defined by the average of the data associated with the component.
- **matrix of variance** Σ_k: representing the dispersion of the data around its center.
- **proportion of** π_k **mixture**: defining the fraction of the data belonging to this class.

If we integrate both sides of 2.10 with respect to x and note that both p(x) and the individual Gaussian components are normalized we obtain:

$$\sum_{k=1}^{K} \pi_k = 1. \tag{2.11}$$

Also, the requirement that $p(x) \geq 0$, together with $\mathcal{N}(x \mid \mu_k, \Sigma_k) \geq 0$, implies $\pi_k \geq 0$ for all k. Combining this with the condition 2.11, we obtain $0 \leq \pi_k \leq 1$.

2.2.3.5 ***Constraints on the parameters of the model***

A priori knowledge on the observations (which should be classified), allows to fix a priori constraints on the parameters of the model of mixture. For example, if the user knows that each class contains equal observations, the proportions of mixture must be equal. For the matrices of variance, the constraints relate to the geometry of the components. Four types of constraints exist:

1. **spherical linear model**: Σ_k is in the form of $\Sigma_k = \sigma^2 I$ where σ is free and I is identity matrix.
2. **diagonal linear model**: Σ_k is in the form of $\Sigma_k = diag(\sigma_1^2, \ldots, \sigma_d^2)$ where $(\sigma_1^2, \ldots, \sigma_d^2)$ is free and $diag(\sigma_1^2, \ldots, \sigma_d^2)$ is diagonal matrix.
3. **general linear model**: $\Sigma_1 = \ldots = \Sigma_k = \Sigma$, where Σ is a positive definite symmetrical matrix.
4. **quadratic model** : no constraint on the matrices of variance Σ_k is posed.

Posing a priori constraints on a model of mixture makes it possible to better estimate the parameters $\theta = (\mu, \sigma)$ and to avoid a singularity problem. A singularity is obtained when the number of free parameters is too large compared to the numbers the observations to classify. In such a case, estimating of the parameters is not reliable.

2.2.3.6 ***Decomposition of the matrices of variances***

Banfield and Raftery [10] proposed a generalization of the various types of matrices of variance previously presented. They used a spectral decomposition of the matrix, which was redefined then by Celeux and Govaert [31] in order to simplify certain calculations for the estimate. The spectral decomposition of the matrix of variance of the class k is defined by:

$$\Sigma_k = \lambda_k D_k A_k D'{}_k \tag{2.12}$$

where :

- $\lambda_k = \mid \Sigma_k \mid^{1/d}, \lambda_k \in \mathbb{R}_+^*$

Σ_k	distribution	Number of parameters
λI	spherical	$a+1$
$\lambda_k I$	Spherical	$a+K$
λB	Diagonal	$a+d$
$\lambda_k B$	Diagonal	$a+d+K-1$
λB_k	Diagonal	$a+kd-(K-1)$
$\lambda_k B_k$	Diagonal	$a+kd$
λC	General	$a+b$
$\lambda_k C$	General	$a+b+K-1$
$\lambda D A_k D'$	General	$a+b+(K-1)(d-1)$
$\lambda_k D A_k D'$	General	$a+b+(K-1)d$
$\lambda D_k A_k D'_k$	General	$a+Kb-(K-1)d$
$\lambda_k D_k A_k D'_k$	General	$a+Kb-(K-1)(d-1)$
λC_k	General	$a+Kb-(K-1)$
$\lambda_k C_k$	General	$a+Kb$

Table 2.1 – Parametrization of the covariance matrix

- D_k is the orthogonal matrix of the vectors of Σ_k
- $A_k = diag(a_{1k} \dots a_{dk})$ is the diagonal matrix of the normalized eigenvalues with the a_{ik} classified by decreasing order on the diagonal and $\mid A_k \mid = 1$.

The advantage of this decomposition is its uniqueness and its facility of interpretation from the geometrical point of view: λ_k represents the volume of the class, D_k its orientation and A_k its form. We can group the 14 models of matrices of variance in three big families:

1. **spherical:** the matrix of class A is the matrix identity and $A = I$. The matrix of orientation D does not have any impact on the parameter.
2. **diagonal:** the matrix $B = DAD'$ is diagonal, D is a matrix of permutation of the canonical base common to all the components.
3. **general:** this family contains all the other models. We note $C = DAD'$.

Table 2.1 summarizes the various models of matrices and details the number of free parameters for each configuration. The number of free parameters emphasize the complexity of each model.

2.2.4 Optimality criteria and algorithms of the parameter estimation for GMM

Now we need to estimate the parameters of a mixture of Gaussians. A general technique for finding the parameters of a GMM is the Expectation-Maximization (EM) algorithm. We begin our discussion by considering the problem of finding the clusters in a set of data points which use a non-probabilistic technique called K-means algorithm [109]. In fact K-means algorithm correspond to a non probabilistic limits of EM applied to mixture of Gaussians and as we shall see, before applying the EM algorithm we often use k-means algorithm to initialize the parameters in a Gaussian mixture model. We continue then by introducing the EM-algorithm which is a generic technique for finding maximum likelihood estimators in latent variable models. The discrete latent variables can be interpreted as defining assignment of data points to specific components of the mixture. At the end we will introduce the MAP learning which solve the ill-posed problem in EM algorithm.

2.2.4.1 ***K-means clustering***

Suppose we have a data set $x_1, \ldots, x_N$ which consists of N observation of a random d-dimension Euclidean variable x. Given the value of K , our goal is to partition the data set into K. The center of cluster k is represented by μ_k. We want to assign the data points and a set of vectors μ_k to the clusters such that the sum of squares of the distances of each data point to its closest vector μ_k is a minimum. We define a set of binary variable $r_{nk} \in \{0, 1\}$ where $k = 1, \ldots, K$ as indicator for each data point x_n. These variables indicate that the data point x_n is assigned to the which of the K clusters. If the data point x_n is assigned to the cluster k, the $r_{nk} = 1$ and $r_{nj} = 0$ for $j \neq k$. Now we define an objective function as follow:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \parallel x_n - \mu_k \parallel^2 \tag{2.13}$$

The goal is to find the values of $\{r_{nk}\}$ and $\{\mu_k\}$ which minimize J (the sum of the squares of the distances of each data point to its assigned vector μ_k). We can do this through an iterative procedure. In each iteration involves two steps corresponding to optimization with respect to the r_{nk} and the μ_k. In the first step we keep the μ_k fixed and minimize J with respect to the r_{nk}. In the second step we keep r_{nk} fixed and minimize J with respect to the μ_k. These two optimization stages will be repeated until convergence is achieved. The first phase can be shown by the following formula:

$$r_{nk} = \begin{cases} 1 & if \quad k = \arg\min_j \parallel x_n - \mu_j \parallel^2 \\ 0 & otherwise \end{cases} \tag{2.14}$$

and the μ_k can be find in the second phase using the following formula:

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} \tag{2.15}$$

Convergence of the algorithm is assured as in each phase, it reduces the value of the objective function J. However it may converge to a local minimum rather than a global minimum of J. The convergence properties of the K-means algorithm are studied in [115]. These two phrases are corresponding E (expectation) and M (maximization) steps of the EM algorithm which we describe next.

2.2.4.2 ***Maximum likelihood criterion and EM algorithm***

Now we point out the definition of the maximum likelihood and then we detail the Expectation-Maximization algorithm ([47, 61, 134]) for estimation the parameters of a GMM. EM is a method most employed to estimate the parameters of a mixture model.

Definition of the likelihood and maximum likelihood

Suppose n independent observations from X, the density of $x_1, \ldots, x_n$ can be expressed by the product of all the marginal densities:

$$L(\theta|X) = p(X \mid \theta) = p(x_1, \ldots, x_n \mid \theta) = \prod_{n=1}^{N} p(x_n \mid \theta). \tag{2.16}$$

This density is called likelihood of the parameters θ. If we have hidden variables $Z = \{\mathbf{z}_n\}_{n=1}^{N}$ then the data likelihood is :

$$L(\theta|X) = p(X \mid \theta) = \prod_{n=1}^{N} \int p(x_n, \mathbf{z}_n|\theta)\, d\mathbf{z}_n \tag{2.17}$$

and if the hidden variables are discrete, the integral is replaced by a sum.

Estimating of the parameters by maximum likelihood is one of the important methods for finding the parameters in many problems. It consists in choosing the parameters of the model which maximize the data likelihood, with respect to the parameter θ (generally the logarithm of likelihood is used instead of likelihood):

$$\theta_{ML} = \arg\max_{\theta} \ln L(\theta|X). \tag{2.18}$$

When we have some variables hidden, integral appears inside the logarithm so the maximization problem becomes difficult and in many practical problems is intractable:

$$\theta_{ML} = \arg\max_{\theta} \sum_{n=1}^{N} \ln \int p(x_n, z_n|\theta)\, dz_n. \tag{2.19}$$

In [47] the EM algorithm was formalized by introducing a distinct arbitrary auxiliary distribution $q_{\mathbf{z}_n}(\mathbf{z}_n)$ over each hidden variable and $q_Z(Z) = \prod_{n=1}^{N} q_{\mathbf{z}_n}(\mathbf{z}_n)$ (from the fact that data X is identically and independently distributed). This algorithm maximize the lower bound with respect to $q_Z(Z)$ while keeping θ fixed (E-step), and then with respect to Θ while keeping $q_Z(Z)$ fixed (M-step).

The maximization problem in the E-step is easily obtained by assigning the posterior distribution of related latent variable to each $q_{\mathbf{z}_n}(\mathbf{z}_n)$:

$$E - step : q_{\mathbf{z}_n}(\mathbf{z}_n) = p(\mathbf{z}_n|x_n, \theta). \tag{2.20}$$

For M-step we choose θ such as the expected value of complete data likelihood will be maximized. The complete data likelihood is defined as :

$$L_c(\theta|X, Z) \equiv p(X, Z|\theta) = \prod_{n=1}^{N} p(x_n, \mathbf{z}_n|\theta) \tag{2.21}$$

and so M-step is as follows:

$$M - step : \theta \leftarrow \arg\max_{\theta} \mathbb{E}_Z\{\ln L_c(\theta|X, Z)\} \tag{2.22}$$

If there is no closed form solution for the M-step, it is sufficient to choose the M-step such that we assure an increase in log-likelihood at each iteration (instead of maximizing it).

In the following, we formulate a Gaussian mixture model in terms of discrete latent variables. This provide us with a deeper insight into this important distribution, and will also serve to motivate the expectation-maximization algorithm.

Mixture of Gaussians in terms of discrete latent variable

Introduction of latent variables allows complicated distributions to be formed from simpler components. The Gaussian mixture model discussed earlier can be interpreted in terms of discrete latent variables. We represent this data set as an $N \times d$ matrix X in which the n^{th} row is given by x_n^T where d is the dimension of data. Similarly, the corresponding latent variables will be denoted by an $N \times K$ matrix Z with rows $\mathbf{z}_n^T$ where K is number of components. The form of Gaussian mixture distribution is governed by the parameters π, μ and Σ, where we have used the notation $\pi \equiv \pi_1, \ldots, \pi_k$, $\mu \equiv \mu_1, \ldots, \mu_k$ and $\Sigma \equiv \Sigma_1, \ldots, \Sigma_k$. In continue, we use the precisions $\Lambda \equiv \Lambda_1, \ldots, \Lambda_k$ where $\Lambda_k = \Sigma_k^{-1}$, instead of Σ as it is more comfortable for our computations. One way to set the values of these parameters is to use maximum likelihood. For the calculation reasons we generally uses the Napierian logarithm of probability. From 2.16 the log of the likelihood function is given by :

$$\ln L(\Theta|X) = \ln p(X \mid \pi, \mu, \Lambda) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Lambda_k) \right\} \tag{2.23}$$

and as a result, there is no closed-form solution for the maximum likelihood of the parameter.

Another way is to formulate the GMM in terms of discrete latent variables. We suppose that we don't know by which component a data point has been generated (the component label related to each data point is unobserved). Using the expectation-maximization algorithm, the expected labels can be found.

Consider the latent variables $z_{nk} \in \{0, 1\}$ that show each data has been generated by which component. If data x_n has been generated by component k, z_{nk} is equal to 1, otherwise it is equal to 0. Latent variables z_{nk} are shown by the set of vector $Z = \{\mathbf{z}_n\}_{n=1}^{N}$. From the definition of z_{nk} we have:

$$\sum_{k=1}^{K} z_{nk} = 1 \qquad \forall n. \tag{2.24}$$

We can write the prior distribution of the latent vectors as equation 2.25 and the conditional distribution of observed data as equation 2.26:

$$p(\mathbf{z}_n|\Theta_N) = \prod_{k=1}^{K} \pi_k^{z_{nk}} \tag{2.25}$$

$$p(x_n|\mathbf{z}_n, \Theta_N) = \prod_{k=1}^{K} \mathcal{N}(x_n|\mu_k, \Lambda_k)^{z_{nk}} \tag{2.26}$$

The result of marginalizing $p(x_n|\Theta_N)$ over the latent variables is the same as equation 2.10:

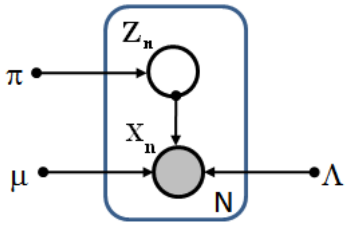


Figure 2.2 – Graphical representation of GMM with corresponding latent variables

$$p(x_n|\Theta_N) = \sum_{n=1}^{N} p(x_n, \mathbf{z}_n|\Theta_N) \tag{2.27}$$

$$= \sum_{n=1}^{N} p(x_n|\mathbf{z}_n, \Theta_N)p(\mathbf{z}_n|\Theta_N) \tag{2.28}$$

$$= \sum_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}} \mathcal{N}(x_n|\mu_k, \Lambda_k)^{z_{nk}} \tag{2.29}$$

$$= \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Lambda_k) \tag{2.30}$$

Figure 2.2 shows graphical representation of the GMM.

Maximum likelihood (ML) learning for GMM

Estimating the parameters by maximum likelihood is one of the most used method. In ML the parameters Θ is chosen such that the model minimize $L(\Theta|X)$. The estimators obtained with this method present good statistical properties:

1. they usually converge towards the true values of the parameters.
2. they are asymptotically without skews and Gaussian.

But maximum likelihood has also some limitation. If we take as many components as observations and each data is associated with the center of a component, likelihood diverges towards the infinite if the variance of each component tends towards zero. Besides, the optimum of likelihood is seldom realizable analytically and requires to resort to iterative methods. In this part we discuss finding the maximum likelihood of the parameters of GMM using EM algorithm.

The EM algorithm use an iterative two steps algorithm. In the first step (E-step) it computes the posterior probability for the latent variables. In the second step (M-step) it maximizes the expected data likelihood with respect to the model parameters.

Now we first show the E-step in which the expectation is taken with respect to the posterior distribution. The joint distribution $p(x_n, \mathbf{z}_n|\Theta_N)$ factorizes which result the posterior probability of the latent variables also factorizes. Using Bayes' rule we obtain:

$$\bar{\rho}_{nk} = p(z_{nk} = 1|x_n, \Theta_N) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Lambda_k)}{p(x_n|\Theta_N)} = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Lambda_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Lambda_k)} \tag{2.31}$$

$\bar{\rho}_{nk}$ are called responsibilities. Now we compute the M-step. The complete data likelihood of the Gaussian mixture model as follows:

$$\ln L_c(\Theta_N|X, Z) = \ln \prod_{n=1}^{N} p(x_n, \mathbf{z}_n|\Theta_N) \tag{2.32}$$

$$= \ln \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}} \mathcal{N}(x_n|\mu_k, \Lambda_k)^{z_{nk}} \tag{2.33}$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk}\{\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Lambda_k)\} \tag{2.34}$$

since the complete log-likelihood is not a marginal probability, the logarithm is inside the sum which leads to a simple formula for maximum likelihood. We obtain the expected complete data log-likelihood as:

$$\mathbb{E}_Z\{\ln L_c(\Theta_N|X, Z)\} = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}_Z\{z_{nk}\}\{\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Lambda_k)\} \tag{2.35}$$

We find $\mathbb{E}_Z\{z_{nk}\}$ (the expectation of the indicator or latent variables with respect to their posterior distribution) as follows:

$$\mathbb{E}_Z\{z_{nk}\} = \sum_{z_{nk}} z_{nk} p(z_{nk}|x_n, \Theta_N) = \sum_{z_{nk}} z_{nk} \frac{\pi_k^{z_{nk}} \mathcal{N}(x_n|\mu_k, \Lambda_k)^{z_{nk}}}{p(x_n|\Theta_N)} \qquad = \bar{\rho}_{nk} \tag{2.36}$$

which is equal to the responsibilities.

The update rules for the component means, precisions and weights can be achieved by maximizing the expected complete data log-likelihood (2.35) respect to the model parameters(Setting the derivatives of $\ln p(x \mid \pi, \mu, \Sigma)$ in 2.35 to zero with respect to the μ_k , Λ_k and π_k):

$$\mu_k = \frac{\sum_{n=1}^{N} \bar{\rho}_{nk} x_n}{\sum_{n=1}^{N} \bar{\rho}_{nk}} \tag{2.37}$$

$$\Lambda_k = \left\{ \frac{\sum_{n=1}^{N} \bar{\rho}_{nk}(x_n - \mu_k)(x_n - \mu_k)^T}{\sum_{n=1}^{N} \bar{\rho}_{nk}} \right\}^{-1} \tag{2.38}$$

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \bar{\rho}_{nk} \tag{2.39}$$

Notice that 2.37 and 2.38 are weighted averages of the responsibilities.

The Expectation-Maximization algorithm for GMM

Now we summarize EM-algorithm for a Gaussian mixture model. As we said, the procedure operates iteratively in two stages. In the E-step, the responsibilities (2.31) are evaluated, while the current model parameters are kept fixed. Next, during the M-step, the model parameters are updated according to the equations 2.37 to 2.39 using the responsibilities variables achieved in the E-step.

We can summarize the EM algorithm for GMM as :

1. initialize the initial parameters θ_0 (the means μ_k , precisions Λ_k and mixing coefficients π_k) and compute the initial value of log-likelihood.
2. **estimation stage:** Evaluate the responsibilities using the current parameter values:

$$\bar{\rho}_{nk} = \frac{\pi_k \mathcal{N}(x_n \mid \mu_k, \Lambda_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n \mid \mu_j, \Lambda_j)} \tag{2.40}$$

3. **maximization stage:** using the responsibilities achieved in E-step, re-estimate the parameters. If we suppose that matrices of variance are full type and free proportions of mixture, estimating of the parameters are achieved from the following calculations:
   - the centers:

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \bar{\rho}_{nk} x_n \tag{2.41}$$

   - the matrices of variance:

$$\Lambda_k^{new} = \left\{ \frac{1}{N_k} \sum_{n=1}^{N} \bar{\rho}_{nk} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T \right\}^{-1} \tag{2.42}$$

   - mixture proportions:

$$\pi_k^{new} = \frac{N_k}{N} \tag{2.43}$$

   where

$$N_k = \sum_{n=1}^{N} \bar{\rho}_{nk} \tag{2.44}$$

   N_k can be interpreted as the effective number of points assigned to cluster k.

4. compute the log-likelihood and check if the convergence of either the parameters or the log-likelihood is achieved. If the convergence criterion is not satisfied return to step 2.

The property of convergence of probability was studied in [195]. It is shown that under general conditions, the values of likelihood converges towards a stationary value. This value is nevertheless a local minimum of probability. The optimum obtained depends on the initial model. The convergence of likelihood is generally fast. Its complexity is in $O(K.d^2.N.E)$, where E is the average number of iterations of algorithm EM. It is thus linear according to the number of data.

Algorithm EM is dependent on manually fixed parameters: the number of components and initial parameters of the model. Compared with the K-means algorithm, EM-algorithm takes many more iteration to reach convergence. Also, each cycle needs significantly more computation. In order to find a suitable initialization, it is common to run first the K-means algorithm that is sequently adapted using EM. The fractions of data point assigned to the respective clusters by K-means can be assigned to mixing coefficient, and the sample covariances of the cluster found by the k-means algorithm can be set as initial covariance matrices. There exist also more elaborate techniques for being less sensitive to the initial conditions of the GMM include stochastic EM (SEM) by Celeux and Diebolt [30] and split-and-merge EM (SMEM) by Ueda et al. [180]. There are also several approaches to efficient learning that have been proposed for high dimensions learning [125, 120, 37]. In [9] it divides the data space into low-dimensional subspace and learns a separate mixture model for each subspace and then it creates a tree of modules to combine the models and produce a single density estimate.

2.2.4.3 ***Maximum a Posteriori(MAP) learning***

We used EM-algorithm in previous section for maximizing the data likelihood, but sometimes it can not find the best possible model for a given data. In fact in EM algorithm, the maximization may fall to the ill-posed problem when there exist a small number of training data (since the likelihood is unbounded). In [73] maximum penalized likelihood or maximum a posteriori (MAP) is investigated to improve the quality of the estimators. It defines some prior knowledge on the problem to constrains the data likelihood. This regularization cause to smooth the model which cause to avoid overfitting and poor local maxima of the unconstrained likelihood. Besides as in this way, it adds a penalization term to the objective function, it makes it usually more concave.

We consider that we have prior information on the range of Θ which implies that for the model parameters Θ, we have the prior distribution $p(\Theta)$. After observing the data X, Bayes' rule allows us to update our prior belief and find the posterior belief of the model parameters:

$$p(\Theta|X) \propto p(X|\Theta)p(\Theta) \tag{2.45}$$

For estimation of the parameters using MAP, we should maximize the log-posterior distribution of the parameters:

$$\begin{aligned}\Theta_{MAP} &= \arg\max_{\Theta} \ln p(\Theta|X) &(2.46)\\ &= \arg\max_{\Theta} \ln L(\Theta|X) + \ln p(\Theta) &(2.47)\end{aligned}$$

The incomplete data likelihood $L(\Theta|X)$ is defined in 2.17. We have the same problem as in ML learning. We can use again the EM algorithm for this problem [73]. The E-step is unchanged because the penalty term only depends on Θ. In fact, the incomplete log-posterior can still be lowerbounded using Jensen's inequality. But the M-step can be computed in the following way:

$$M - step : \Theta \leftarrow \arg\max_{\Theta} \mathbb{E}_Z\{\ln L_c(\Theta|X, Z)\} + \ln p(\Theta). \tag{2.48}$$

If the EM algorithm estimate $\hat{\Theta}_{MAP}$ as the values for Θ_{MAP}, the predictive distribution in MAP learning can be obtained by:

$$p(X) \approx p(X|\hat{\Theta}_{MAP}) \tag{2.49}$$

In [138], the MAP framework is applied to the GMM. They chose the conjugate to the GMM as the priors. A prior $p(\Theta)$ is said conjugate to the density $p(x|\Theta)$, if the posterior of $p(\Theta|x)$ have the same functional form as $p(\Theta)$. It is useful when we want the same parametric form of the prior distribution transfer to the posterior distribution.

As it is mentioned for example in [68], the conjugate prior on the the mixture proportions of GMM is a Dirichlet distribution:

$$D(\pi|\alpha) = c(\alpha)\prod_{k=1}^{K}\pi_k^{\alpha_k-1} \tag{2.50}$$

and the Gaussian-Wishart distribution can be used as conjugate prior on the mean and the precision of a single multivariate Gaussian component ([68]):

$$\mathcal{N}W(\mu_k,\Lambda_k|\Theta_{NW_k}) = \mathcal{N}(\mu_k|m_k,\beta_k\Lambda_k)W(\Lambda_k\ w_k,\nu_k). \tag{2.51}$$

By choosing these types of priors, the penalized data log-likelihood will be obtained as follows:

$$\ln L_{MAP}(\Theta_N|X) = \ln(p|X|\Theta_N) + \ln p(\Theta_N) \tag{2.52}$$

$$= \ln L(\Theta_N|X) + \ln D(\pi|\alpha) + \sum_{k=1}^{K}\ln\mathcal{N}W(\mu_k,\Lambda_k|\Theta_{NW_k}) \tag{2.53}$$

where we have denoted the joint prior on the parameters of the GMM by $p(\Theta_N)$ which now is equal to:

$$p(\Theta_N) = D(\pi|\alpha)\prod_{k=1}^{K}\mathcal{N}W(\mu_k,\Lambda_k|\Theta_{NW_k}). \tag{2.54}$$

Here again, the penalized log-likelihood cannot be maximized directly (like ML). However, as before we can define the responsibilities:

$$\bar{\rho}_{nk} = \frac{\pi_k\mathcal{N}(x_n|\mu_k,\Lambda_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x|\mu_j,\Lambda_j)}. \tag{2.55}$$

For the maximization step we keep these quantities fix and find the updating rules for the component means, precisions and weights:

$$\mu_k = \frac{\sum_{n=1}^{N}\bar{\rho}_{nk}x_n + \beta_k m_k}{\sum_{n=1}^{N}\bar{\rho}_{nk} + \beta_k} \tag{2.56}$$

$$\Lambda_k = \left\{\frac{\sum_{n=1}^{N}\bar{\rho}_{nk}(x_n-\mu_k)(x_n-\mu_k)^T}{\sum_{n=1}^{N}\bar{\rho}_{nk} + w_k - d} + \frac{\beta_k(\mu_k-m_k)(\mu_k-m_k)^T + \nu_k}{\sum_{n=1}^{N}\bar{\rho}_{nk} + w_k - d}\right\} \tag{2.57}$$

$$\pi_k = \frac{\sum_{n=1}^{N}\bar{\rho}_{nk} + \alpha_k - 1}{N + \sum_{j=1}^{K}\alpha_j - K} \tag{2.58}$$

In summary, MAP learning of the GMM consists of computing the responsibilities (E-step) and maximizing the resulting penalized log-likelihood (M-step) in an iteratively manner.

2.2.5 Model Selection

Supposing a classification context, the number of component in a model is not known, so it is necessary to determine it in some way, in order to be able to provide an automatic system. Several techniques exist to determine and compare several solutions (models) in a space of assumptions and select the parameters which represent the structure of the data as well as possible. It is important to choose a method to select the model which represent the real structure of the data as well as possible.

We present here two general methods: method by re-sampling, penalization by the measurement of quality of a model. Then we present some numerical criteria to determine the complexity of a mixture model.

2.2.5.1 ***Method by re-sampling***

Several work [102, 124, 170] assume that the structure of the data can be found from a subset of the data. They propose to determine the complexity of a mixture model by comparing the estimates of the parameters on several samples of data obtained from the initial unit. This re-sampling method is related to statistical simulation, known as Monte Carlo [111]. It is based on the techniques of generation of artificial samples from only a part of data and is used when the analytical complexity of the problem does not allow classical inference. It consists in repeating analyzes on various samples of data simulated then study the fluctuations of the results.

The basic difference between the Monte Carlo method and that by re-sampling is that in the first the data are artificial, while for the second simulations are based on real data. Several methods of validation make it possible to create samples of data in various manners: the permutation [60], cross validation [100], Jackknife [146] and bootstrap [54].

Permutation method consists in creating a sample from a permutation of the initial data. The cross validation divides the whole of the initial data into several subsets of variable size. Jackknife proposes to compare several subsets of data obtained by removing only one data of the initial unit (this technique is generally used to detect the isolated data). Finally bootstrap consists in simulating m samples of the same size n as the initial sample. They are obtained by pulling randomly (certain data have a high weight and others a null weight).

The method by re-sampling proposes to compare the estimate of the parameters of the mixture model on several samples of data obtained with one of the preceding methods. It is necessary to define an error function to compare the various tests which makes it possible to obtain the relevant model to represent the structure of the data.

2.2.5.2 ***Penalization by the measurement of quality of a model***

Many models has a criterion to evaluate its quality: for example the quadratic criterion of error for a hierarchical algorithm or probability for a mixture model. The approach consists in finding a compromise between such a criterion and the simplicity of the partition. We present two methods, the penalization by complexity and the detection of a curve of quality.

Penalization by complexity

These approaches are based on the principle of Occam which consists in selecting the simplest model for describing the data well. It is the problem of finding a compromise between the quality of the model and its simplicity. If we base only on a quality criterion (or error) of the partition, the model will be more complex and we obtain a model adapting perfectly to the structure of the data, but presenting little interest. The criterion of likelihood or the quadratic criterion of error are optimized for an increasing number of components. A model including n observations will present a measurement of maximum quality for a model of n classes: each class is associated with an observation. Such a result is of course not relevant and since the classification obtained does not have any more interest, our objective has the ability to summarize the observations. The objective of this approach is to penalize the criterion of quality by the complexity of the model (the number of free parameters).

In the context of mixture models, several numerical criteria were proposed in the literature, penalizing probability by the number of free parameters of the model: criteria AIC (Akaike Information Criterion [2, 3]) and AIC3 [24, 25] which we will present them in section 2.2.5.3.

Practically, to compare two models, it is enough to calculate the numerical criterion for each one of them and then select the model which optimize it. This technique makes it possible to evaluate models of different complexities using a function, generally quickly and easily calculable. The comparison between two models is thus possible in a direct way. On the other hand to obtain the " best" model, it is always necessary to test a whole of models.

Method by curve of error

These approaches consist in detecting a significant change of slope in a curve of error which decreases as the complexity increases. The complexity of the model obtained at the bent of the curve, is supposed to provide the model which represent the structure of the data as well as possible. These approaches do not have a theoretical base but seems rather empirical: the idea is to increase the complexity of the model and to detect when the variation of the error, according to complexity, becomes weak. It can select, like the preceding method, a compromise between complexity and the measurement of quality.

It is necessary to draw the curve with the number of components in the model in X-coordinate and the error value for each model tested in ordinate. The algorithm of classification to obtain the parameters of the models is free but it is preferable to use a hierarchical algorithm (agglomerative or by division). That avoids recomputing the whole of the parameters to pass from the model of k components to that to $k \pm 1$ (since in the case of the hierarchical algorithms, it is based on the parameters of the model obtained with iteration $i-1$). As the error function is free, we can choose any metric evaluations for it. The hierarchical approach is also preferable to obtain the error criterion of a model since it can be obtained starting from the preceding model (by calculating the variation of error pulled by the fusion or the division of components). An example of criterion of error can be quite simply the likelihood of the data or the Kullback-Leibler distance.

Once the curve of error obtained, the objective is to detect a curve indicating a significant change of variation. The slope of the error curve is supposed being strongly decreasing for the first models of low complexity, this will be explained by a great difference between the models. For the models which are too complex, the variation of error becomes much weaker since the models are increasingly similar. The changes of slope in the curve reveals a bend in the curve, related to the supposed model to have best complexity to represent the structure of the data.

Several techniques exist to determine the bent:

- the greatest amplitude between two points;
- the first point with the second derivative lower than a static limit;
- the point with the greatest second derivative ;
- the point more far from one line obtained from the curve.

Other more complex methods exist, for example work of El Salvador [161], seeking two lines to summarize the two significant parts of the curve of error and choosing their intersection to determine the number of components in the model. Methods similar to research of a bent exist [176, 177], by transforming the error curve into a new function, and selecting the minimum or the maximum to determine the adequate model.

2.2.5.3 ***Bayesian criteria and approximation***

We present here the various existing numerical criteria which rise from the Bayesian theory and more classical statistical approaches.

Bayesian approaches

This approach consists in retaining a posteriori on the model M. This probability is expressed by:

$$P(M \mid X) \propto P(X \mid M)P(M) \tag{2.59}$$

If all the models are equiprobable a priori, it amounts selecting the model optimizing $P(X \mid M)$. This probability, called integrated probability or marginal probability, is defined by:

$$P(X \mid M) = \int_{\Theta} \mathcal{N}(X \mid \theta, M)P(\theta \mid M)\, d\theta \tag{2.60}$$

where $\mathcal{N}(X \mid \theta, M)$ is the density of mixture of the model M and $P(\theta \mid M)$ the priori model of θ under this model. To calculate this probability, it is necessary to define priori model on θ and to evaluate the integral. The analytical evaluation can not easily be solved, but various approximations exist [94]: numerical methods, methods of Monte Carlo, or the approximation of Laplace-Metropolis (see [131] for the two last). An alternative way consists in approaching marginal log-likelihood by the maximum of log-likelihood penalized by a function of the number of free parameters and number of data as a vector X. The obtained expression is simpler and makes it possible to be freed to define priori model. This approximation is based on the factor of Bayes, which makes it possible to estimate the relationship between two integrated probabilities.

The factor of Bayes makes it possible to introduce criteria BIC and AWE. The criteria BIC [165] (Bayes Criterion Information) and AWE [10] (Approximate Weight of Evidence) are two approximations of the factor of Bayes. The factor of Bayes is obtained from the criterion of Schwarz [165]. The model selected is that which maximizes the factor of Bayes. That amounts choosing the model which minimizes the BIC criterion:

$$BIC(M) = -2L(M) + q(M)\ln(n) \tag{2.61}$$

where $L(M)$ is the maximum of log-likelihood with the model M, $q(M)$ the number of free parameters in this model. Criterion AWE is based on a different approximation from the factor of Bayes. Contrary to the BIC criterion, this criterion supposes the constraints a priori on the parameters of the Gaussian model known and is limited to the choices of the number of components. Then, we obtains criterion AWE:

$$AWE(k) = -2CL(k) + 2q(k)(3/2 + \ln(n)) \tag{2.62}$$

which $CL(k)$ is the maximum of log-likelihood classifying for k classes. The selected model is that which minimizes this term.

Finally, the Baysian approach presents a bond with the theory of coding MDL. A criterion similar to the BIC criterion is the criterion MDL (Minimum Length Description) [153] which define as :

$$MDL(M) = -L(M) + q(M)\ln(n) \tag{2.63}$$

This criterion presents a penalization with fewer number of free parameters than for the BIC criterion. It should be noticed that these criteria are supposed being effective only for samples of data of big size.

Penalization of probability by the complexity of the model

The preceding criteria amount penalizing likelihood by the complexity of the model and the number of data. We present in this section a generalization of this type of criterion. Likelihood can be interpreted like a measurement of quality of the adjustment of the density to the data. This criterion nevertheless presents the disadvantage of choosing always the most complex model. Criteria penalizing likelihood by the complexity of the model were thus proposed to regulate the problem. They aim to find a compromise between the likelihood of a model and its complexity. Their general standard is:

$$criteria(M) = -2L(M) + g(M), \tag{2.64}$$

where $g(m)$ is a function making it possible to calculate the complexity of a model $M \in H$. We select the model which minimizes this criterion. The complexity of a model depends in general on the number of free parameters which it contains. The function g is thus in the form

$$g(m) = \tau q(m) \tag{2.65}$$

where $\tau \in \mathbb{R}$ and $q(m)$ is the number of free parameters in the model M. The parameter τ thus makes it possible to regulate the compromise between the probability and the complexity of the model. The problem is to determine the value of τ. Let us note that its optimal value can not exist since complexity cannot be a linear function of the number of parameters. Criteria baysian BIC and AWE clarified previously have a form identical to the equation 2.64. Nevertheless their theoretical base makes it possible to provide a value to the parameter τ . Notice that it is a function of size n observations. We present now criteria AIC and AIC3. Criteria AIC and AIC3 are very much used since they have the advantage of being simple to obtain.

Criterion AIC (Akaike Criterion Information) [2, 3] is historically the first criterion of penalization of this kind. It is defined by:

$$AIC(M) = -2L(M) + 2q(M) \tag{2.66}$$

The τ term is equal to 2. This choice was justified in [3, 4] by two reasons. $AIC(M)$ is an estimator of the average risk to choose the estimator of the maximum of likelihood under the model M. This risk is calculated while based on the Kullback-Leibler distance between the true distribution and the distribution with the parameters estimated by the maximum of likelihood. The comparisons of models by AIC are asymptotically equivalent to those found on the factors of Bayes. But other research prove that this choice for the value of the parameter τ is debatable [108]. Criterion AIC is obtained based on the classical theory of the test of assumptions. This criterion is not thus theoretically relevant to determine the number of components in a model. To solve this problem, an alternative of AIC, called AIC3 [24, 25], was proposed with $\tau = 3$:

$$AIC3(M) = -2L(M) + 3q(M) \tag{2.67}$$

The penalization with the free parameters of the model is stronger here than for criterion AIC.

2.2.6 Techniques for model combination

For improving performance, we can often combine together the multiple models. It can be useful specially in the distributed systems in which many models may exist for one class as there are different data and algorithms on such systems. The combining might be done by training L different models and then for prediction, average the predictions of all models. This approaches often are called committees. One of the important types of committees is boosting in which it trains some models in sequence and in each stage for training a particular model, it use the error function which depend on the performance of the previous models. Mixture of experts and hierarchical mixtures of experts are two other approaches which can be used to fuse the learning models.

2.2.6.1 ***Boosting***

Boosting is a technique for combining a number of base classifier to produce a powerful classifier whose performance is better than of any of the classifiers. In boosting, the base classifiers are trained in sequence. For training each classifier, a weighted form of the data set is used. The coefficient (weight) associated with each data point depends on the performance of the previous classifiers. The points which are misclassified in one of the classifiers are given greater weight for training the next classifier. When all the classifiers have been trained, their predictions are then combined through a weighted majority voting scheme.

The algorithm AdaBoost (adaptive boosting) is most widely used form of boosting which has been introduced in [65]. It can solve many of the practical difficulties of the earlier boosting algorithms. One of the main ideas of this algorithm is to find a set of weights over the training data set $X = \{x_1, \ldots x_n\}$. The job of the base learners (which are also called weak prediction rules or rules of thumb) is to find a base classifier $h_t : X \to R$. If we have T base classifiers $(h_1, \ldots, h_T)$, the set of weights are the distribution D_t. In fact $D_t(i)$ is the weights of the distribution on training example i for the classifier t. All the weights set equally at first. AdaBoost repeatedly calls the base learning algorithm in a series of rounds $t = 1, \ldots, T$. On each step, the incorrectly classified examples in previous step are found and the weights for these examples are increased. This cause the base learner to focus on the hard examples in the training set. The simplest case is when the range of output of each classifier is binary, i.e., restricted to $\{-1, +1\}$. In this case the input of this algorithm is a training set $(x_1, y_1), \ldots, (x_n, y_n)$ where the $x_i \in X$, and y_i is a label in $Y = \{-1, +1\}$ and the base learner's job is to minimize the error. When the classifier h_t has been trained, AdaBoost assign a value $\alpha_t \in R$ to it which shows its importance. It update also the distribution D_t. The final or combined classifiers H is a weighted majority vote of the T base classifiers where α_t is the weight assigned to h_t

2.2.6.2 ***Mixture of experts***

In the adaptive mixture of local experts [84] there is much emphasis to making the experts local so we expect much performance over all the system. A basic level of locality is gained by focusing each expert on its distribution to have maximal performance, instead of giving it the compensate for errors of other experts. Further localization is achieved if for each pattern, we give the higher learning rates to the better performing expert. In mixture of experts we try to improve the performance by assigning different subtasks to different learners. In this setting several "expert" classifiers (or regression estimators) and a gating function are joint and train concurrently. Based on the current input, a probability value is assigned to each of the experts by the gating function. The mixing coefficients are also functions of the input variable, so that

$$p(t|x) = \sum_{k=1}^{K} \pi_k(x) p_k(t|x) \tag{2.68}$$

in which individual component densities $p_k(t|x)$ are the experts and the mixing coefficients $\pi_k(x)$ are the gating functions. The usual constrains for mixing coefficients is $0 \leq \pi_k(x) \leq 1$ and $\sum_k \pi_k(x) = 1$ which must be satisfied by the gating function $\pi_k(x)$. In this technique, we suppose that each expert can make predictions in its own region so the distribution in different regions of input space are modeled by different components. The component which is more dominant for each region is chosen by the gating function.

When the system works in the training stage, the probability of a pattern appearing in the experts'training set is given by the mixing coefficient. But when it works in the test stage, the mixing coefficients shows the relative contribution of each expert to the ensemble. There are two goals in the training step: (i) for a given expert, find its optimal gating function, (ii) for each gating function, train the correspondent expert system to achieve the best performance on its region.

2.2.6.3 ***Hierarchical mixtures of experts***

The mixture of expert use linear models for the gating and expert functions which cause a significant limitation for it. The hierarchical mixture of experts [90] is a much more flexible model which uses multilevel gating function or HME model. In this architecture, the experts may be built by a combination of lower level experts and gating functions in a a tree format. The gating networks sit at the nonterminals of the tree and the expert networks sit at the leaves of the tree. When the vector X is given as input, each expert produces an output vector for that. These output vectors send to the up of the tree and in each level they are blend by the gating functions. In [91], the EM algorithm was used for training the HME.

2.2.6.4 ***Comparison of boosting and mixture of experts***

The mixture of experts is suitable if we can divide the pattern to simpler and homogenous subsets and in each of these subsets, the learning task is not difficult. However, in real-world problems the required gating function may be complex, even when such a partition exists. The main advantages of the mixture of experts are (i) localizing the different experts and (ii) using a dynamic model to combine the outputs.

In boosting, in each stage, the distributions are selected in the way that force the classifier to become an expert on the pattern which previous classifiers has an error. It is proper for difficult patterns while maintaining a reasonably good performance on easier patterns. Its limitation is that it can not be applied to unlabeled data.

2.3 Content-based multimedia information retrieval

Multimedia data which can be combination of data such as texts, audio, images and video, is quite different from structured data and semi-structured (text) data. These types of data are media-specific (with specific operations), possibly very huge, and described by metadata. As the use of digital media increases, effective retrieval and management techniques become more important. The aim of multimedia data management is to provide high-level capabilities for searching and manipulating multimedia collections efficiently and accurately. Such techniques are required to facilitate the effective capturing, storing, searching and browsing of large multimedia databases.

The traditional and common methods of multimedia retrieval usually utilize some method of adding metadata such as captioning, keywords, or descriptions to the multimedia documents which allows the media to be indexed based on keyword annotations and accessed by text-based searching [58]. Textual information about multimedia can be easily searched using existing technology, but requires humans to personally describe every image in the database. This method of keywords based indexing and retrieval has many limitations especially in the context of multimedia databases [50, 167]:

- first, it is difficult to describe some media contents in words, for example we cannot easily express an image having complicated texture patterns or a piece of melody in textual.

- second, it is time-consuming and expensive to manual annotation of text phrases for a large database. Besides, the difficulty finding desired information will increase as the number of media in the database grows. It becomes infeasible to manually annotate all attributes of the media content for example annotating a 30-minute video which contains more than 50,000 images will consume a vast amount of time and is very expensive.

- third, it is almost impossible to capture all concepts, thoughts and feelings for the content of any media which make difficult to describe it with a complete set of key words specially when users may have different interests in the same multimedia object.

- finally, even if all relevant object characteristics are annotated, there are different users which have different indexing languages and vocabularies. It is also possible to miss images that use different synonyms in their descriptions.

To overcome these difficulties, we can use the contents of media for locating the desired data in a large database rather than annotation entered by the human. This method is called content based retrieval (CBR) which automatically index the content information [72]. There have been extensive studies on the design of automatic content-based indexing and retrieval systems. These contents may include color, shape and texture for visual media and phonemes, pitch, rhythm, cepstral coefficients for audio/speech data. In fact, we can use whatever that can distinguish the classes of data. A feature dataset contains the feature vectors of the media. In a typical content-based retrieval system, the contents of the media in the database are extracted and described by multi-dimensional feature vectors, which are called also descriptors. In this thesis, we search the classes besides the features vectors.

Classically, GMM has been used in content based multimedia (speech,image and video) recognition applications. GMM is used as a generic probabilistic model for multivariate densities which can represent arbitrary densities. This makes GMM a suited approach for unconstrained text-independent applications. In [149, 151, 156] they described the use of GMMs for text-independent speaker identification. We can find also an extension of GMM-based systems to speaker verification in [148, 150]. Vasconcelos and Lippman [185], conducted a comparison among various image representation schemes in the context of probabilistic content-based indexing and retrieval. In this study, they show that the Gaussian mixture model outperforms other image representations like the standard representation for texture-based retrieval, color histograms and color correlograms (Huang et al. [83]). They mentioned some of the advantages of the GMM over the other representations including reducing number of parameters in the image representation and the usage of a smooth basis function (the Gaussian). Using GMM, the basic functions are placed (in the feature space) in the regions which are more populated. The result is that the resources is not allocated for modeling regions of the space which have zero probability. This leads to reduce the number of parameters needed and keep the complexity low. There are very efficient procedures for building indexing structures using GMM since it cluster the Gaussians instead of points (for example in [183]). For representing video content also the recent works include statistical models. Each frame can be represented via models such as the GMM model in feature space. In [56], GMM is used to model the spatio-temporal events. In [74] another approach for video representation is described which create a piece-wise GMM framework clustering on space-time regions in feature space and can detect the video events. In this approach they first try to extract of video segments using unsupervised clustering and via GMM then they extended it to a piecewise GMM framework (instead of a single global GMM model) which enables to proceed online instead batch mode.

In a CBR, users submit query examples to the retrieval system to retrieve desired class data. The system first represents these examples using feature vectors then it computes the similarities or distances between the feature vectors of the query example and the feature vectors of the media in the feature dataset. For an efficient searching of the media database in retrieval, the system use an indexing approach. Finally, the search results may be ranked and the top search results (which are most similar to the query) are returned. A general Content-based retrieval is illustrated in figure 2.3.

Research and development issues in CBR cover a range of topics. Some of the most important are: extracting the features from raw multimedia documents, finding suitable ways for describing media content, providing compact storage for large multimedia databases, matching query, efficiently accessing by content to the stored multimedia , designing usable interfaces for

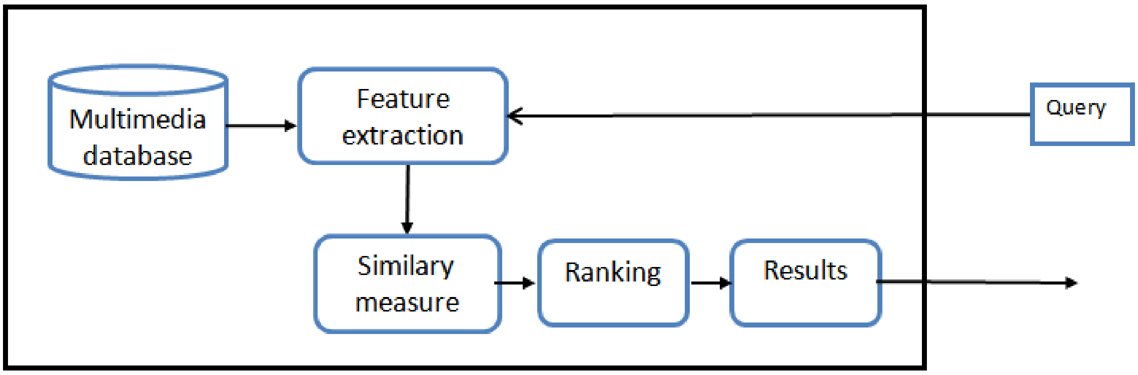


Figure 2.3 – Simple architecture for general content-based retrieval

CBR systems and etc.

The retrieval systems and media analysis have received widely the public and media interest. We can hope that in the near future, many other domains also use the technologies related to the retrieval systems. In future media retrieval, both text-based and content-based search will be used. Although from a user point of view, the text-based is considered more reliable but there is an immense potential to combine the two to make the media search engines with ability to access the part of multimedia documents which are hidden now in the Web.

One of the subfield of CBR is content-based image retrieval (CBIR) which aims to avoid the use of textual descriptions in images and retrieves them based on their visual similarity to a query image or user-specified image features. The term 'content' in CBIR refer to any information which can be derived from the image. It seems that the term content-based for image retrieval used for the first time in [95] which performed an automatic retrieval of images by color and shape features. This term has since been widely used for retrieving images based on their features that can be automatically extracted. There are a wide range of application which can use CBIR technology [76] for example ”medical diagnosis” , ”the military”, ”crime prevention” and ”Web searching”. Riya, a public domain search engine, has been recently developed with the ability of searching pictures of people and products on the Web using image retrieval and face recognition techniques. CBIR technology is also being applied to the divers domains as family album management, remote sensing , astronomy, mineralogy, and botany [44, 139, 164, 199]

In the following we present general techniques for CBIR.

2.3.1 Content based Image retrieval (CBIR)

To model a CBIR, we usually need to do the followings:

- extracting the features.
- construction of the mathematical representation (signatures) from features.
- finding similarity measure based on the signatures.
- clustering and classification.

Now describe each of the above steps.

2.3.1.1 ***Feature Extraction***

One of the important issues before any kind of reasoning about the contents of the media is the extracting useful information from the raw data (such as recognizing the presence of particular textures or shapes). To find the approaches that can effectively code the attributes of the media, we should find the features which are most useful for representing its contents.

A feature extraction function is defined as follow: Let $DB = \{I_i\}_{i=1}^{n}$ be a set of objects in the database. With a set of feature parameters $\theta = \{\theta_i\}_{i=1}^{n}$, a feature extraction function f is defined as:

$$f : I \times \theta \to R^d \tag{2.69}$$

In all content-based retrieval multimedia database, users may want to retrieve objects similar to a query based on some kinds of features. Therefore, the useful features of the multimedia object should be extracted, when one is inserted to the database. These features is transformed to the vectors and organized for content-based retrieval. There are two types of features:

- low-level : The features such as color, shape, texture in images, object motion in video and loudness, power spectrum, bandwidth and pitch in sounds are called low-level. They are mainly derived from the media and does not refer to any external semantics [49]. These types of features can be usually obtained by signal processing techniques and can answer queries for example "find images in which the distribution of blue is more than 30%" which might retrieve many images with blue sky. Many approaches using low-level features are developed for various purposes [58, 75].
- high-level features: high-level features also called semantic features use more complex analysis techniques such as pattern recognition or statistical classification. Some of examples are instruments, timbre, rhythm and events which involve different degrees of semantics. Processing of high-level queries are difficult since the system should have external knowledge with the description of low-level features, that is called semantic gap. For minimizing semantic gap, two solutions have been proposed. The first is automatic metadata generation. The second is relevance feedback which allows the retrieval system to understand the semantic context of a query.

Usually, we need to determine an optimal set of features. The principal component analysis or some other feature selection techniques can be used for this mean (for example in [85, 122]). The acquired feature space is generally of high dimension and the statistical methods are usually more suitable and reduce the dimension.

2.3.1.2 ***Construction of signatures from features***

In the content based retrieval we should investigate about has two intrinsic problems: (a) how to describe an media mathematically (b) finding a similarity measurement to compare a pair of media based on their descriptions. The mathematical description which is used for media retrieval is called its signature.

As the most exploited type of image signature are region-based signature we focus more on it and its mathematical connection with histograms. However, there are other forms of signatures such as distributions extracted from a collection of local feature vectors like a continuous density function [51], or a spatial stochastic model [105].

Both histograms and region-based signatures can be supposed as sets of weighted vectors. These sets are equivalent to discrete distributions when their weights sum up to one. Histogram was used widely in early works as a form of distribution. Suppose $x_{i,j} \in R^d$, denote the feature vectors in the d-dimensional Euclidean space and R^d is divided into fixed bins. To have a histogram, the percentage of $x_{i,j}$'s of each bin is calculated. If there are k bins the histogram treats as a k-dimensional vector $(f_1, f_2, ..., f_k)^t$, where the frequency of the l-th bin is denoted as f_l. In [157] there is a discussion of the disadvantages of treating histograms simply as vectors of frequencies.

Now we talk about the essentially of region-based signature, as used in [48]. Consider the data set $\{x_{i,j}, 1 \leq i, 1 \leq j\}$. For grouping the feature vectors $x_{i,j}$ into k clusters, a clustering procedure like k-means is applied to the data set such that the mean of $x_{i,j}$'s in the cluster l be z_l. In fact, we achieved a summary of the data set as $\{(z_1, w_1), ..., (z_k, w_k)\}$, where w_l is the percentage of $x_{i,j}$'s grouped into cluster l. The pixels (i, j) which are in the same cluster forms a relatively homogeneous region. This way is widely used to segment images and the signatures $\{(z_1, w_1), ..., (z_k, w_k)\}$ are called region-based. The region-based signature provides a compact description of images because it adapts the representative vectors z_l to the images. In [48] there is a discussion about the region-based and their efficiency for retrieval.

2.3.1.3 ***Defining similarity based on the signatures***

Once we made the decision about the choice of image signatures, the next problem is how to use them for accurate image retrieval. There has been a large number of fundamentally different frameworks proposed in the recent years.

The choice of distances and similarity measure depend on the mathematical representation of the signature. For example, region based signatures obtain by segmenting images using local feature vectors and summarizing these feature vectors. For these types of signatures, the definition of distance between sets of vectors can be used as similarity. However, it is not obvious as defining distance between single vectors.

Another example is vector quantization (VQ) on image blocks which generate codebooks and use these codebooks for representation and retrieval. The segmentation is not critical for these approaches. We can estimate the distributions of the feature vectors as the signatures.

For these methods Kullback-Leibler (K-L) divergence have been proposed in [51, 118] as measure of similarities. The K-L divergence, is an asymmetric measure of difference between two distributions $f(.)$ and $g(.)$, and defined as

$$K(f, g) = \int_{-\infty}^{+\infty} f(x) \log \frac{f(x)}{g(x)} dx, \qquad K(f, g) = \sum_x f(x) \log \frac{f(x)}{g(x)} \tag{2.70}$$

in the continuous and discrete cases respectively. It is also known as the relative entropy.

Some authors have noted when images are represented as single vectors there are difficulties for measuring by metrics in the feature space. However, it may be find a non-linear manifold of image vectors and use the geodesic distance instead of Euclidean distance. In fact the visual perception may corresponds better with this non-linear subspace than the original linear space.

There are also a number of probabilistic frameworks for CBIR [88, 188]. In [188] the feature selection, feature representation, and similarity measure are integrated into a combined Bayesian formulation to minimize the probability of retrieval error and then in [184] the complexity of this approach is reduced by approximately modeling the probability distribution of the image features using VQ.

2.3.1.4 Clustering and Classification

When a query is sent to a retrieval system, it is not good idea to compare it to all of media documents in the database for retrieval. Instead we organize the media documents into groups or classes and perform queries at the class level instead of image level. Thus, first the class that best explain the query will be found and then the images in that class will be searched to find the best match. There are two general ways for grouping: classification and clustering. When we have labeled data, classification can be applied for assigning unseen images into broad categories. It may also be used for automatic annotation. If there is no labeled data, we can use unsupervised clustering to speedup the retrieval and improve the result visualization.

The similarity measures have an important natural role in image clustering. Clustering usually is based on the optimization of a measure of the clustering quality. An example is k-means clustering in which the measuring tool is the sum of distances between every vector in the cluster and the center of cluster. But in image categorization there are many methods which does not require similarity metrics. Only when we have a query for the images in a large database the images which belong to the same class may use step of similarity measurement. In this case the similarity matching is followed by classification step to retrieval the queried image.

In [36] it is proposed to use the similarity information among the retrieved images for the CBIR system which the retrieved images are ranked only by the relevance to the query image. For example, for improving retrieval process, the clusters can be dynamically generated, to be proper specifically to the query image.

One way of clustering is statistical modeling in which each cluster is treated as a pattern and each cluster is specified by a distribution. The whole data set is a mixture of these distributions. The Gaussian distribution is mostly used for continuous data and thus a mixture of Gaussians is fit to the data set, usually by the EM algorithm [119]. Center locations and shapes of the clusters are determined by the means and covariance matrices of the Gaussian components. Mixture modeling approach provides a partition of data described by an estimated density, which sometimes is itself desired [51].

2.3.2 Different class descriptions problem in content based retrieval

In previous section, we presented the content-based retrieval using features. The features mainly focus on the content of media but they can be used to design the class representations. In this thesis, we work on the classes beyond the feature vectors. There may be different descriptions of a unique class. A closely problem is how to find a better description from the existing different descriptions. There are some works for this mean such as mixture of experts which tries to improve the classifiers. We don't talk here about combination of different features to have a better feature such as discussed for example in [168] for the case of musical retrieval. We assume the different descriptions of a class for examples created by probabilistic models. This can be occurred when there are different training data for a unique class or different algorithms for finding the descriptions or models for the classes. Schema of such problem is shown in fig 2.4.

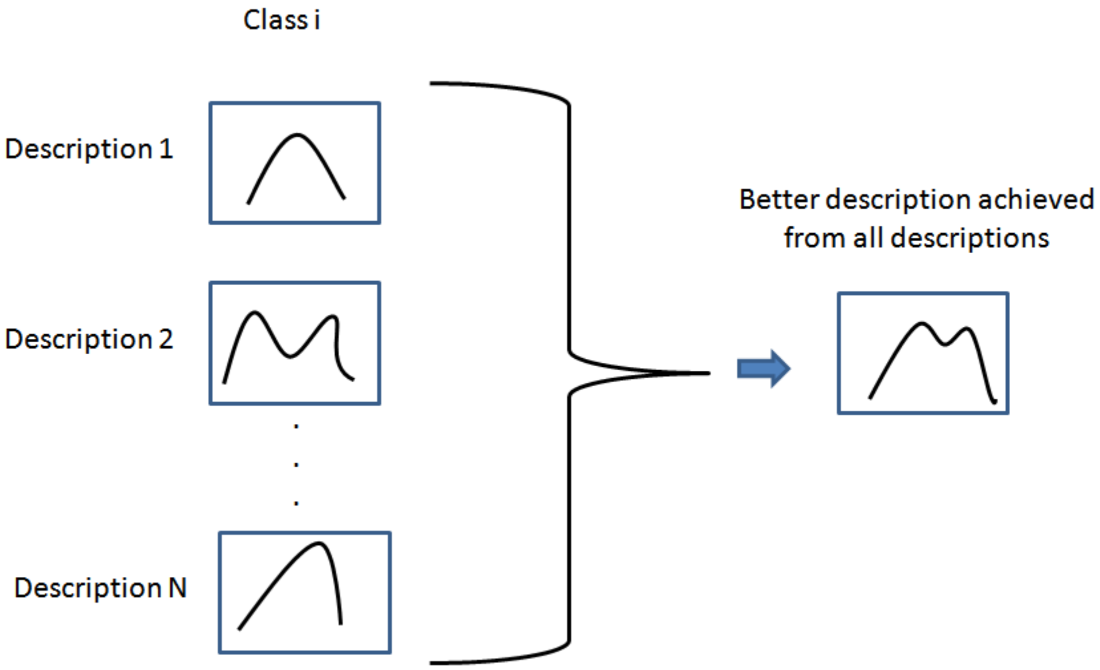


Figure 2.4 – Class conditional description

In such case, the problem is in fact how to find the best description among different existent description or finding better description by some way of combining them.

2.3.3 Query techniques

Query is an ability for users by which the user express his needs and allow conceptual access to multimedia. The query system is crucial in any retrieval system. Query is the searching for a set of results using examples which have similar contents. The common queries for multimedia databases is similarity search that is based on a similarity measure or distance function. Queries in content-based retrieval systems depend on the media type. For example in video and image retrieval systems we may send query by sketch or query by painting and for audio retrieval the query by singing may be used. Interacting with the system to provide relevance feedback is a powerful technique that allows the user to choose the better search results based on their relevance. In the case of image databases, a useful alternative to querying is browsing which organizing an image collection in a suitable way. The browsing approach is allowed in more interactive and iterative searching. Sometimes a query involving two or more features that may be decomposed into sub-queries and their results will finally integrated. In the following we talk about important types of querying.

2.3.3.1 ***Query by example***

Query by example is to provide a sample desired for output and asking the system to retrieve the examples of the same kind. All of the result images should have the common elements in the example. The search algorithms are different and depend on the application.

There are two options for providing example images:

- having a pre-existing image.
- drawing a rough approximation of the image which is searched (For example using general shapes or colors).

However, it is not possible to have always an image example. There exist alternative query methods for this mean which most are based on the ideas of QBIC (the system developed by IBM) [62]. In the original QBIC, users could specify colors as queries either by choosing the relative amounts of red, green and blue or by selecting a color from a palette. As colors does not depend on the image size or orientation this way is one of the most widely used techniques. For searching in this way, the system usually compare the color histograms. Other way is sending the query by specifying the texture from a palette where the system look for visual patterns in images. Shape queries are also possible by sketching the object [101]. In this approaches, we usually use the shape of a particular region. The shapes often can be determined by segmenting the image or applying edge detection which sometimes are difficult and may require user intervention to have more accurate results. In some shape retrieval approaches the ability, for sketching shapes are limited to primitives such as rectangles and circles [166, 169] and in some other like the one that proposed in [33] the sketch-based interfaces are developed. There exist also composite querying methods which provide the users more flexibility for expressing the desired object. Sketch-based retrieval for color images is discussed in [32]. As in the 2 dimension image queries, it may not possible to capture some arrangement of objects, in [6] the query for 3 dimension models is motivated.

2.3.3.2 ***Semantic retrieval***

In the ideal CBIR system the user should have ability to send the semantic queries such as finding the images of someone or specified dog. This type of tasks are very difficult for computers. The specified person may not always be in the same position. Therefore most current CBIR systems use lower-level (like color,shape and texture) or higher level (like face) features. There are also systems which are designed for a specific domain and limit semantics.

In a text-based search, a query containing a specified word may results millions of Web pages. The smart search engines try to rank these results and return the best matches by analyzing the query. An alterative CBIR approach to rating is tackled in [45] which tries to find the semantically relevant images. This approach is based on re-defining the goal as a model that can characterize aesthetics in general by study the photo rating trends in public photo-sharing communities.

An alternative approach is to use image annotation as a supervised categorization problem. One of the earliest attempts at image annotation can be found in [104]. The system, ALIP (Automatic Linguistic Indexing of Pictures) [106] takes a semantic categories and in order to capturing inter-scale and intra-scale spatial dependencies of the image features, it uses an approach based on a 2-D multi-resolution hidden Markov models. The system learns and stores independently the models for individual classes. In the annotation step, the system calculates the likelihoods of the query image for each learned model/category then it chooses in a statistical manner, the annotations which correspond to the most likely categories. ALIPR, is the version of ALIP that is designed for real time image annotation. Another approach is introduced in [34] in which a confidence label is given to the images for each of trained semantic labels using Bayes Point machines. When a keyword is searched the relevant images will be ranked using these confidence labels.

2.3.3.3 ***Relevance feedback***

The relevant feedback is originally developed for texts [160] in order to improve the effectiveness of information retrieval systems. It is the ability to refine the searches by having online interaction with the user. Its main idea is the understanding of the user's needs. For the first time this ability is demonstrated within a keyword-based system in [145], but it has been successfully implemented in some CBIR systems for example in [159, 169].

In the image systems, the retrieval system returns initial image results in response to a query, based on predefined similarity measures. Then the user should identify the positive examples which are relevant to the query. The system analyze the feedbacks using a learning algorithm and refine the results. [154] and [158] are two learning algorithm for updating iteratively the weight estimations. Many algorithms are proposed in recent years for relevance feedback. In [198], a semantic feedback based is proposed in which the feedbacks are based on image semantics. These semantics are image labels which are defined manually.

We can also find the approaches which use probabilistic models. For example the PicHunter system [41] use a distribution over a series of potential goals to show the uncertainly of user's goals and then by applying the Bayes' rule, the target image is selected. In [172], a Gaussian density is used for the features of the positive examples. After each feedback the system re-rank the images useing a Baysian classifier. The approach proposed in [187] suppose a prior on the user's intent. When new information (user feedback) is entered, the system use the Bayes' rules to find the new prior for the next feedback.

2.3.4 Indexing

For rapidly access to multimedia data when a query is requested, we need an efficient indexing method specially in large systems that millions of pictures and keywords may be stored. Indexing, in fact, is concerned with the physical access to multimedia data but visual and acoustic have low level representations and can not support the high level semantic queries such as finding words spoken in an audio or individuals in an image. To find the easy approaches for searching media documents in such databases, they must be processed to extract either semantic or perceptual information that is useful for indexing and searching, for instance, color, shape, and texture (CST) vectors in image and pitch, rhythm or phoneme transcription in acoustic. The multimedia features are usually modeled as points in a multidimensional space and can be efficiently indexed using multidimensional index structures. The two categories for such indexing structures are treebased and hashing-based. The treebased indexing methods also classified into two main classes:

- rectangle-based : these methods use rectangles to partition the features into groups for indexing. R-tree, R+-tree, R*-tree, and SR-tree are some classical examples of rectangle-based indexing methods.

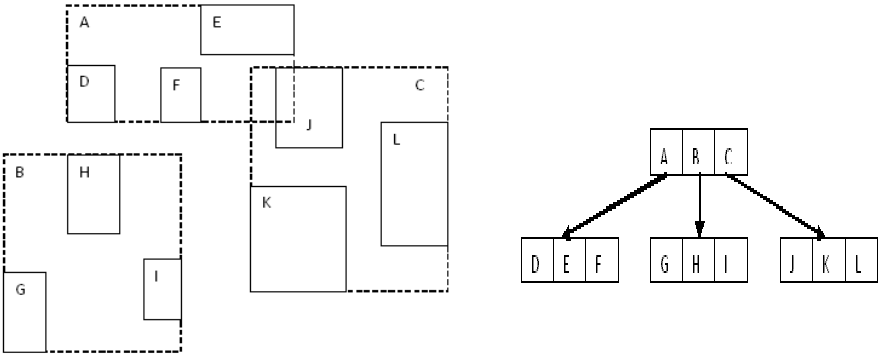


Figure 2.5 – (a) An input data set partitioned by using minimum bounding rectangles. (b) The corresponding R-tree structure.

- partition-based indexing: these methods use lines or curves to organize the feature vector space into partitions for indexing. Quad-tree, k-d tree, VP-tree, and MVP-tree are some classical examples for partition-based indexing methods.

R-tree [78] and R*-tree [14] are two popular multi-dimensional indexing methods which can perform well with a limit of up to 20 dimensions. Many other approaches for indexing multimedia uses these two indexing methods, for example in [140] R-trees are used for indexing images represented as attributed relational graphs (ARG) and in [130] wavelet coefficients are used for image representations and R-trees for indexing.

In the following we talk about R-tree as a rectangle-based indexing and Quad-tree as a partition-based indexing.

R-tree : R-tree [78] is the generalization version of the B-tree [12, 39] for multi-dimensional data indexing. It uses rectangles to organize data into groups. The partitioning continue in hierarchically manner until all the leaf nodes contain a number of instances within a pre-defined range. R-tree is a balanced tree with leaf nodes and non-leaf nodes only. The root node has at least two children unless it is a leaf node. Figure 2.3.4 shows an example of R-tree.

Let the maximum and the minimum number of entries that a node can contain, is respectively $M$and $m < M$. Then, every leaf node except the root node must contain a number of entries between m and M. Also, every non-leaf node except the root has between m and M children. For building R-tree data objects are inserted one by one starting from the root node using a minimum bounding rectangle (MBR). MBR is the smallest rectangle containing all the data objects for the node. Data continue to be inserted into the node until overflow. In this time, the corresponding rectangle is partitioned into several smaller rectangles for child nodes, using a splitting algorithm. Deletion is other a major operation of R-tree. When a data object is deleted from a node, if the deleted node contains less than m objects, a merging algorithm is applied. The searching algorithm for R-tree is not very difficult. During a query, all the nodes in R-tree that have MBRs overlapped with the query rectangle are searched in order to find the result of the query.

Quad-tree: Quad-tree [59] is one of the first indexing methods developed for multi-dimensional data. It is the generalization version of binary tree. In Quad-tree the feature vector space are divided into partitions according to the direction of the data points. For example in two-dimensional space, each non-leaf node in has four child nodes representing its four directions(NE, SE, SW, NW). k-d tree [15] is also a kind multi-dimensional binary search tree where k denotes the dimensionality of the search space. The searching algorithm is similar to quad-tree except we just need to compare one different attribute value at each level of tree. For example, in 2-D space, we are comparing the x coordinates at even levels while we are comparing the y coordinates at odd levels.

2.3.5 Video retrieval

Video sequences are a form of image data which increase important quickly. Many researchers have tried to adapt the CBIR techniques for video retrieval.

They have worked on different approaches, but most of them used remarkably similar ways [5]:

- the first step is usually dividing the video sequences into individual shots which is determined by a sudden change in seen image content and camera angle. The changes can be found by analyzing the color histograms, texture and motion vectors.
- from each shot, a keyframe is selected which is a representative for that shot(again, by analysis of color and texture).
- after providing a complete set of keyframes, they can be manually annotated, or stored in an image database for content-based retrieval or browsing.

In recent years, one of the important needs in video data is the detection of ”semantic concepts” and ”event recognition” which needs to segment the video into meaningful units. The main difficulty for finding these dynamic contents is to express them in terms of low-level video features such as color, texture, motion and shape but sufficiently discriminant regarding their content [192, 197].

One way is to classify video sequences into different genres by introducing statistical models for video structure components which has been studied in [189]. If we consider specific motion classes, for example human motion, rigid motion or dynamic textures we can use related motion models. Some examples are explicitly derived from kinematic laws (e.g., 3-D kinematic screw for rigid motion [57], articulated motion for human motion [67]), conveniently represented by simple parametric models (e.g. AR models for dynamic textures [52], 2-D polynomial models for camera effects [21]).

Using Gaussian mixture models for representing color features and space-time locations has been presented in [74]. For detecting dynamic regions, the authors provide a space-time segmentation of the video. In [28], in order to characterize motion, the authors used wavelet coefficients which has been estimated from image sequences then they used the derived models for indexing tasks. In [133], first the motion descriptors are determined by obtaining the temporal slices from the image sequences then these motion features and color features were combined and using the K-means classification technique, video shots have been grouped.

In [142, 143], two different probabilistic motion models are defined: original and parsimonious. Probabilistic motion models both for the dominant image motion (camera motion) and the residual image motion (scene motion) are further recombined for the event detection task. For each class of events, probabilistic motion models have to be learnt. A training set of meaningful and pre-identified events of video samples is used to learn the motion models (both for camera motion and scene motion). Finally, to restrict the recognition issue to a limited and pertinent set of classes, they designed a two-step event detection method. In the first step, candidate segments that have potential interest are selected. The second step is a classification problem and tries to recognize the relevant events (in terms of dynamic content) among the selected segments.

2.4 Distributed computing

A distributed computing system consists of spatially separated processes that do not share a common memory and communicate with each other by passing messages over communication channels. It is primary focus on controlling idle CPU cycles and storage space of many of networked systems that work together on a particularly processing problem. However there are some limitations for the growth of such processing models such as lack of proper applications, bandwidth bottlenecks, management, security problems, and standardization challenges. In recent years, there are new interests in this idea, since there are more advances in some of the related branches, for example peer-to-peer networks and gossip-based protocols. Advances in network technology, like peer-to-peer networks, have highlighted the requirement for designing efficient algorithms and applications for large amounts of distributed data. Gossip-based protocols also called epidemic algorithms are one of the important subjects in communication technology in which each node repeatedly contacts some other node in random and exchanges information with them. They can be used also for distributed computing such as node aggregation.

2.4.1 Definition

Distributed computing is a type of parallel computing. Parallel computing relate to the development of programs to perform a common task using cooperating multiple concurrent processes. It is often said to two or more processors on the same computer that run simultaneously the different parts of a program. In more general, distributed computing is a computer processing technique in which different components and objects related to an application may be located on different computers that can communicate each other using a network. The computers may have different hardware components and different file systems type. The separated processes communicate asynchronously by passing messages over communication channels and usually does not have a common memory. One of the important requirements for distributed computing is to define a set of standards for communicating each other.

In a local networks the expression ”cluster computing” may be used for a distributed computing while nowadays the ”grid computing” is used in wide-area networks. ”BOINC” is an example of distributed computing in which large problems are divided as a some of some small problems and then distributed to many computers.

2.4.2 Various types of distributed systems

There exist many types of distributed systems, such as clusters, grids, P2P networks, distributed storage systems. A cluster contains some interconnected computers which can be imaged as a single super-computer. Clusters are usually used in high performance scientific engineering and business applications. In a grid distribution system, coordinated sharing and aggregation of distributed are possible. Grids are generally used for supporting the applications emerging in e-Science and e-Business technologies, that mainly include the geographically distributed communities of people who collaborate to solve large scale problems. These activities usually need to share some resources such as computers, data, applications and scientific instruments. P2P networks are distributed systems that are mainly decentralized and allow the users to have applications such as file-sharing, online multiuser gaming, instant messaging, and content distribution over public networks. Distributed storage systems like NFS (Network File System) allow users to have a unified view of data stored on different computers and file systems that can be on different or the same networks.

We can divide distributed programming into one of several basic architectures or categories:

- client-server: client program contacts the server for data, displays it to the user in a proper format.

- 3-tier architecture: three tier systems are a type of client server, but the client intelligence is at a middle tier. In these systems user interface, computer data storage and data access are developed and maintained as independent modules which simplifies application deployment. Most Web applications are 3-Tier.

- N-tier architecture: N-Tier refers mainly to the applications which forward their requests to other services. An example is the Web applications that uses middle ware to service data requests between a user and a database.

- tightly coupled (clustered): refers usually to a set of machine which divide a task into some parts and run that in parallel. The result of all parts will back together to make the final result.

- peer-to-peer : refers to the architecture where there is no special machine or machines for managing the network resources or services. In these systems the responsibilities are divided among all computers which are called peers.

The main features of a distributed system include [128]:

- functional separation: based on capability and purpose of each entity in the system its function can be different. It is also possible to extend easily the existing components and add new components.
- inherent distribution: the entities and resources (information, people, and systems) are inherently distributed. For example, different people can use different information which is created in the system. Different systems or applications can generate, analyze , store and use information.
- reliability: replications and backup of information can be set at different locations, so failure of one or more components do not bring down the entire system.
- scalability: the system work efficiently with increasing addition of a resource to increase performance or availability.
- economy: this architecture can help reduce the cost of ownership by sharing of resources by many nodes.

As a result, the nodes in a distributed architecture can operate concurrently and possibly autonomously. They do their tasks independently and communicate by exchanging messages. One of important properties is that the nodes are usually heterogenous (use different hardware, software and etc.). Generally, we like these systems to be fault tolerant and not to depend only on a single node or process which contains the whole knowledge or state of the system. There are various kinds of distributed systems that are used for different kinds of problems.

2.4.3 Peer to peer networks

Peer-to-peer networks are mainly used for connecting nodes using a largely ad hoc connections and depend on the computing power and bandwidth of the participants. These networks are useful for many purposes such as sharing files that are containing audio, video, data. Realtime data can also passed using P2P technology (for example telephony traffic).

In a pure peer-to-peer network, there is no notion of clients or servers. Instead there are equal peer nodes that function as server and client in the same time. This model of network are different from the client-server model where some computers should serve the others.

The nodes in P2P system can join the network from anywhere at any time or leave the system at any time. Each node is connected to a relatively small set of neighbors which in turn is connected to more nodes. P2P systems, become quickly popular due to their properties such as simplicity, scalability, simple administration, fault-tolerance, and self-organizing nature. There is an increasing hope that P2P networks can be used for building large-scale information retrieval systems at low cost [103]. Many P2P search techniques [38, 43, 152] which are usually based on simple keyword matching have already been proposed in recent year. P2P is in fact a logical layer over the internet which makes a new logical network that allows the users to share and access any type of data and services. An important property in the peer-to-peer networks is that all clients may provide resources (including storage space, bandwidth, and computing power). When more nodes join the system the demands increase but the capacity of the system also increases. This is different with a client-server architecture that has a fix set of servers and adding more clients could mean slower data transfer. Another property is robustness that is acquired from replicating data over multiple peers. Such networks are constrained by some characteristics : (i) there is usually no central entity for computation, communication and time-synchronization, (ii) the nodes may not know the network topology (iii) in each time, nodes may join or leave the network, so the network topology may change, and (iv) the computational power of the nodes may be limited. These constraints motivate the design of simple decentralized algorithms for such networks in which each node can exchange information with only a few of its neighbors in a time instance.

*2.4.3.1 **Classification of peer-to-peer networks***

In a P2P system, there is a link between any two nodes which know each other. In the other words, if a node knows the location of other peer, there is a directed edge between them. The peer-to-peer networks can be classified according to their degree of centralization and searching type in three categories:

- **centralized peer-to-peer networks**: in this types of P2P, all functions (such as indexing, query processing, access control) and information about the network are put on a single node. These networks have mechanisms with specialized index nodes and are similar the client/server systems. The central node responds to the requests of the information. in such systems, the clients who wants to send and receive data, should connect directly to this central server. The indices which contains absolute addresses of data are put on the central node. The central server is responsible to handle control flow and data flow and normal peers are responsible for hosting available resources. Simplicity and security are main advantages of centralized systems because all information is concentrated in one node and only one host need to be protected.
- **decentralized or pure peer-to-peer networks**: in this topology, there is no central server or central controller so there is no managing of the network. All nodes connected to each other in a P2P manner. The nodes which may host different applications are in equal level of responsibilities and have the role of clients and servers in the same time. There is no central routing in decentralized P2P networks. One of advantages of pure P2P systems is their scalability. Any node can join a network in each time and start exchanging data with any other node. Another advantage of decentralized systems is their fault tolerant property, as the failure of any node on the network has no effect on the working of the rest of the system. Kazaa is an example of pure peer-to-peer networks.
- **hybrid peer-to-peer Systems**: in a hybrid peer-to-peer system, data flow takes place in a pure peer-to-peer manner but the control information is managed through a central server. The control server monitor the other peers and ensures information coherence. This architecture decrease the problems of managements in pure P2P systems. If the central server fails, the data flow of existing applications can continue between nodes but the system can not affect the changes in data flow. These systems have better scalability than a centralized system but as control information flows through a single node, they suffer from scalability problems for control information.

Some networks such as [129] use a client-server structure for some tasks (for example searching) and a peer-to-peer structure for others. Gnutella [69] or Freenet [137] use a peer-to-peer structure for all purposes.

The P2P can be also classified as unstructured or structured, based on how the nodes are linked to each other:

- **unstructured peer-to-peer** : an unstructured P2P network is the type of P2P that the links are established arbitrarily. In unstructured P2P a new peer can be easily joint. Most of the popular P2P networks such as Gnutella and FastTrack are unstructured. For finding a desired piece of data, a query will be flooded through the network to find as many peers as possible that have data. The main disadvantage of such networks is that there is no correlation between the content of data and the peers who have it. Consequently, there is no guarantee that the queries (which use flooding) can find the peer with the desired data. Other problem is that flooding causes a high traffic on the network.

- **structured peer-to-Peer** : in the structured P2P networks a protocol is used to route efficiently a search query to the peer that has the desired data. This ensure that if the desired data exist on the network (even if it is extremely rare), the query can find it. Such a guarantee, requires more structured network. The common structured P2P networks usually use hash table (DHT) in which a hash value is assigned to each file and determines the peer that contains the file. Some well known DHTs are Chord, Pastry, Tapestry, CAN, and Tulip.

2.4.3.2 ***Split stream multicast system***

There are also approaches for multimedia streaming in P2P. The video/audio stream for P2P is split into several smaller streams. Then each stream is recognize by a sequence number to enable the play back to place it in correct sequence. Different peers that has been joint to the streaming session, exchange their information. A peer can retrieves multimedia data by requesting that from other nodes, and also send its available data to other peers. In [29] an approach was proposed to overcome the problem of unbalanced loads.

2.4.4 Randomized Gossip Algorithms

There was a shift from centralized computation to fully distributed systems in the last decade. One of the most important distributed systems is the large-scale P2P networks which have millions of nodes and many algorithms are developed for distributed information storage and retrieval on these networks. In the other side, the advances in hardware caused increasing of the physical sensor networks that may consist of many number of small sensor nodes. This motivated the design of the aggregation algorithms for such systems with highly fault-tolerant, as nodes and links may fail at each time or temporary communication disruptions may occur. Algorithms for such networks need to be robust against changes in topology, for examples in financial data on the Internet and weather data that observed by a set of sensors.

One way is to consider data as rumor and spread it among all participating nodes using randomized communication. This problem can be supposed as n players that exchange information in parallel. In every round, there are players who generate rumors (updates) that should be distributed among all players. At random times the communications are established between two players and they should exchange the new rumors(new information). The major problem is that the players do not know about the rumors that their partners have already received. Demers et al. [46] introduced the epidemic or gossip algorithms for such problems. They proposed two concepts:

- anti-entropy: in this case, every node chooses regularly another node at random and exchange the complete data base contents.

- rumor mongering: in this case, each node that receives a new information becomes a ”hot rumor”. The state of this node changes to ”hot rumor”, in which it periodically chooses other nodes at random and sends the rumor to the others. There are several mechanisms for deciding when a hot rumor should stop transmission and becomes ”cold”.

It seems that anti-entropy is a very reliable solution but it produces a huge amount of communication. In contrast, in the rumor mongering approach, the nodes exchange only recent updates that reduce significantly the communication overhead. Sometimes a combination of both approach may be used: the mongering approach frequently and anti-entropy very rarely for only ensure that all updates has been distributed to all nodes.

The idea of rumor or gossip was fist introduced as ”push transmission” in which data is sent from the caller node to the called node [46]. Another idea that was presented in [46] is ”pull transmission” in which rumor data is sent from the called to the calling. In a pull approach the number of nodes which has not been received the new information, decreases much faster than push scheme.

After the work of Demers et al., there are many studies on epidemic algorithms for example in [1, 79, 112]. Karp et al. [93] studied the efficiency of these randomized algorithms. In particular, they studied the complexity of time and communication complexity for a simple model randomized communication. They tried to reduce the large communication overhead of epidemic algorithms and showed that this overhead can be decreased. They gave an algorithm with only $\Theta(n \ln\ln n)$ transmissions and $\Theta(\ln n)$ rounds.

Gossip-based protocols are now appeared as an important communication system. The good properties of gossip are extensively reviewed in [55], but we can summarize the advantages of gossip-based protocols as: (i) they can scale to a huge number of nodes; (ii) high stability under stress and disruptions; (iii) they are simple while having strong performance with moderate overhead and don’t need usually error recovery mechanisms.

2.4.5 Node aggregation using Gossip based algorithm

In many large systems, it is more important to have the aggregated values of over whole network than data at each nodes [77, 116, 181] or in other words we wish to compute and disseminate the aggregates of the data. The aggregation, means finding the statistics (for example sum, average, variance, etc.) of a distributed set of values. This is specially required when the size of networks increase that motivate to find a fully decentralized algorithm with minimal processing and communication requirements and good fault-tolerant behavior. Collecting all local data in one specified node make the problem of communication bottlenecks or even storage problem in that node.

Bawa et al. [11] defined the node aggregation problem and restrict the problem to sums, averages, minima and maxima then they present some protocols for this problem that most of them are based on building trees. In such problems, they can be formulated as follows: we have a network with n nodes, on each node i a value x_i (or a set of values) are stored. The goal is to find in a decentralized manner the aggregate function (sums, averages, quantiles, etc.) of the values on all nodes and using small messages only.

Other solution is to use the gossip-based models of computation [86, 96] which provides robustness, scalability and simplicity. In [96] the authors showed how to compute random samples or quantiles using a completely decentralize gossip-based solution. Their approaches also answers to many other aggregation queries in a decentralized fashion. They extend the study of aggregation to sums and averages. They supposed a weaker failure model than that proposed in [11], and found simple protocols for the above problems and showed that their protocols converge exponentially fast to the true answer. However, they observed that the convergence of the protocols is slow when the protocols use flooding on network with slowly mixing random walks such as grid-like graphs. This motivated investigating the techniques for speeding up communication with respect the topology of network.

Boyd et al. [23] presented a framework to analyze and design the randomized distributed averaging algorithms for an connected network by establishing the corresponding optimization problem to be convex. Using this framework, the fastest averaging algorithm can be found. In fact, they used the subgradient method on the structure of the network to solve the semidefinite program in a distributed fashion (as solving semidefinite programs in a distributed manner is not possible). This correspond finding the optimal averaging algorithm. In their algorithms, the network can be started with some arbitrary averaging matrix, (for example one derived from the natural random walk), then it allows selftuning the weights and converge to the optimal weights corresponding to the fastest averaging algorithm. The approach works locally and does not need any central coordination.

The solutions that we presented in above can be used for small data. As the size of the data increases (for example the multimedia data which their size are huge), it is not easy to use such algorithms because the size of data to transfer on the network is huge.

2.5 Distributed estimation of Gaussian mixtures

In previous section, we investigated P2P network and gossip protocol and we presented how they can be used for distributed computing in node aggregation. One of the problems with increasing of distributed data is how to use them for learning the models of different classes. This need is specially felt for the multimedia data which are distributed on such networks. One approach is to gather all data in one node and find the model of a class from whole data related to that class. It may possible for small data or small networks, but for huge data such as multimedia data can not be done easily. In fact in many applications, we are interested in learning a global model from such data like a probability distribution or a clustering of the data, without first transferring all the data to a central node. We may call it ”distributed learning”. For this mean, we need a decentralized algorithm which can compute and disseminate the aggregates of the data to find a model with minimal processing and communication. It should also has good fault-tolerant behavior.

In this section we talks about the existent approaches for distributed learning for a Gaussian mixture model or distributed estimation of Gaussian mixtures. This idea use randomized gossip-based distributed protocol for implementing an EM algorithm in a decentralized manner to estimate a Gaussian mixture models from distributed data.

2.5.1 Gossip based distributed algorithm for Gaussian mixture model

Gaussian mixture models are a rich family of probability distributions that usually use maximum likelihood and the EM algorithm [47] for learning its parameters. They have many applications in statistics, pattern recognition and machine learning [119]. There are some distributed implementation of the exact EM algorithm. For example data sufficient statistics are propagated on the network [64, 135]. Nowak [135] proposed a distributed EM algorithm for estimating the means and variances for the Gaussians of GMM for modeling the measurements on the nodes. In this approach, it is assumed that the measurements of a sensor network are samples drawn from a GMM distribution in which the means and covariances are unknown and the mixture coefficients are different in each sensor. Then it tries to estimate the parameters of the global density. After estimating the global density, each node can determine the component weights related to it by comparing to its local data. They combined this algorithm with an incremental learning scheme as in [132]. This approach use only one node for computations at any time step. This algorithm cycles through the network and use the local data at each node (and the statistics which are passed from the previous node) to perform incremental E and M steps. However, the assumption of a routing mechanism in this algorithm for traversing all nodes in the network is not practical for some of distributed systems.

Kowalczyk and Vlassis [99] proposed a decentralized implementation of the EM algorithm for learning a global Gaussian mixture model from data, for the networks that arbitrary point-to-point communication between nodes is possible. The algorithm named ”Newscast EM”, use a gossip protocol and all nodes are running the same protocol in parallel. In this algorithm, a set of data is assumed to be distributed over the nodes of a network and drawn independently from a common Gaussian mixture model. Newscast EM use a decentralized gossip protocol for computation in the M-step in a number of cycles. The protocol is very robust, scalable, and simple to implement.

Now we look to the Newscast algorithm in more detail. In Newscast EM (NEM), a set of data $\{x_i\}$ are supposed that has been distributed over the nodes of a network (one data point per node). The goal is to learn the parameters $\Theta = \{\pi_s, \mu_s, C_s\}_{s=1}^{k}$ in decentralized manner using as little communication as possible, assuming that the data is samples of a common k-component Gaussian mixture $p(x)$. In other words, the NEM algorithm aims to use an averaging protocol like [86] for estimating the parameters Θ of $p(x)$ using the EM solution described in section 2.2.3.

As the E-step of NEM is the same as the E-step in the standard EM algorithm, all the nodes can perform it in parallel. But M-step is implemented as a sequence of gossip-based cycles. In this step, first a local estimate Θ_i of parameters is initialized for each node i. Then each nodes contacts repeatedly to the other nodes in gossip manner and replace its local models by the one achieved by averaging the two models. Using this approach, the local models of all nodes converge to the global model with the correct parameter Θ. Algorithm 5 is the complete algorithm, that runs on each node. The local estimates of node i for the parameters of component s is shown by $\Theta_i = \{\pi_{si}, \mu_{si}, \tilde{C}_{si}\}$ where the parameter $\tilde{C}_{si}$ is defined such that $C_{si} = \tilde{C}_{si} - \mu_{si}\mu^T{}_{si}$.

The number of total average operation in M-step is $k[1 + d + d(d+1)/2]$ where k is the number of component and d is dimension. To find the EM-correct estimates, for any component s, we should use the weighted averages over all the local estimates :

$$\frac{\sum_{i=1}^{n} \pi_{si}\mu_{si}}{\sum_{i=1}^{n} \pi_{si}} = \mu_s \tag{2.74}$$

and similarly for the $\tilde{C}_{si}$.

As reported in [136], this gossip-based learning algorithm produces the results which are essentially identical to those obtained by the standard EM algorithm, but much faster. The M-step in each node is performed in $O(\log n)$ time, but if all data send to a central node and use an ordinary EM algorithm, it will be $O(n)$. Each node contacts $O(\log n)$ to the other nodes (one per gossip cycle) in each M-step so the complexity of total number of messages sent over the network is $O(n \log n)$.

2.5.2 Gossip-based greedy Gaussian mixture learning

Given a set of training data, EM algorithm can be used for estimating a mixture model for them. EM begin with an initial value for parameters and then optimize the mixture parameters

Algorithm 5 Newscast-EM algorithm

1. Initialization: set $q_i(s)$ to some random positive value and then normalize all $q_i(s)$ to sum to 1 over all s.

2. M-step: initialize i's local estimates for each component s as follows: $\pi_{si} = q_i(s)$, $\mu_{si} = x_i$, $\tilde{C}_{si} = x_i x_i^T$. Then repeat for τ cycles:

a. Contact a node $j = f(i)$ from i's local cache.

b. Nodes i and j update their local estimates for each component s as follows:

$$\pi^T{}_{si} = \pi^T{}_{sj} = \frac{\pi_{si} + \pi_{sj}}{2} \tag{2.71}$$

$$\mu^T{}_{si} = \mu^T{}_{sj} = \frac{\pi_{si}\mu_{si} + \pi_{sj}\mu_{sj}}{\pi_{si} + \pi_{sj}} \tag{2.72}$$

$$\tilde{C}^T_{si} = \tilde{C}^T_{sj} = \frac{\pi_{si}\tilde{C}_{si} + \pi_{sj}\tilde{C}_{sj}}{\pi_{si} + \pi_{sj}} \tag{2.73}$$

c. Nodes i and j update their caches appropriately.

3. E-step: compute new responsibilities $q_i(s) = p(s \| x_i)$ for each component s using the M-step estimates π_{si}, μ_{si}, and $C_{si} = \tilde{C}_{si} - \mu_{si}\mu^T{}_{si}$.

4. Go to step 2, unless a stopping criterion is satisfied that involves the parameter estimates themselves or the energy F.[2]

using an iterative algorithm. In each step, it tries to increase the likelihood function by producing new parameters. The problem is that EM is a local optimization algorithm and may fall into a local maximum. For solving this problem, several initialization methods have been proposed [121].

One way to tackle the problem of sensitively of EM is to use a greedy learning approach in which the components are added one by one until it reach the desired number of components [107, 191]. In this approach, the overhead data a little increased but can produce much better results than standard EM algorithm as reported in [191, 193]. Another similar approach is proposed in [180] in which components use a split and merge way to avoid local maxima.

In the gossip-based EM algorithm also the initialization is random which may cause the same problem of local maxima. In order to remove the problem of sensitive initialization in the algorithm of [99], the greedy learning of Gaussian mixture models has been proposed in [194]. For resolving the problem, they derived a gossip-based greedy Gaussian mixture learning algorithm in which components are added sequentially to the mixture. They used in fact the gossip based distributed implementation of the greedy learning algorithm of [191].

Finally, this technique needs a way to find a proper component to add to the mixture. This requires a randomized function that generate the candidate components. The algorithm thus alternate between two steps: (i) mixture updating using the gossip-based EM algorithm described, (ii) component allocation. If the random function for generating the new components depends on the data as in [191, 193], it can be implemented by gossip-based protocols. In the greedy Gossip-based Gaussian mixture learning algorithm, this type of random function is used, thus, in each component allocation step, the algorithm uses gossip-based parallel search, starting from multiple initial guesses to find the component.

2.6 Conclusion

The applications that use or produce the multimedia data are increase more and more and multimedia documents become an important task in many domains. As a consequent:

- multimedia documents growth rapidly.

- as multimedia indexing and retrieval systems are largely a learning/recognition issues and based on the learning algorithms, we need the proper learning algorithms for multimedia documents. But numerous pattern recognition algorithms are CPU demanding and the multimedia pattern to index is huge.

- meanwhile services on the internet are evolving from centralized client-server architectures to fully distributed and P2P architectures as the nodes become more powerful and provide more services. We can find the huge multimedia data and software and different pattern recognition algorithms on the nodes of the distributed networks but they work mainly locally.

- we need the new algorithms for indexing, searching and etc. for multimedia documents on the distributed networks. One of the important needs is an efficient distributed learning for multimedia documents which are distributed on the network.

In the first three section of this chapter we talked about the three fields (machine learning, multimedia recognition systems and distributed systems) which are three components that motive the need of a distributed learning for multimedia on distributed networks. This is an open problem yet and our goal is to find an approach for this problem. In the distributed networks, each part of data is placed on a node on the network. We like to find a global model using whole data on the network. This will cause having better model which better describe data. This can be done by two types of approaches:

- the approaches that send new data (related to each class) from one node to other node until all nodes have the same data for all classes. So whole class data is placed on all nodes. Thus, a learning algorithm on each node can use the data of that node to make a model for each class. It can be used only for the small data and is not proper for the huge data like multimedia data.

- the approaches which use the data of different nodes to find the models for the classes without transferring data on the network. These approaches can be used for multimedia data.

As Gaussian mixture model has many good properties, thus, in the last section we presented the ”distributed estimation of Gaussian mixture models” and the approaches exist for it. In the next chapter we will propose an approach for distributed learning of Gaussian Mixture model to find a global description of a class.

CHAPTER 3

Low-cost distributed learning of a probabilistic mixture model

As we said in previous chapter, to have better models in a distributed data system, we need decentralized algorithms for learning a global model. This can be performed by two way : (i) aggregating whole data then estimating a model (ii) estimating a global model from distributed data using a decentralized algorithm. The first approach needs transferring data on the network that is not appropriate for multimedia data as multimedia data is huge. In contrast, in the latter approach, there is no need to send the data over network and each node contribute in estimating a global model. In this chapter we present the distributed estimation using aggregation the Gaussian mixture models. This approach can be used for multimedia data which are distributed on a network such as P2P networks.

3.1 Introduction

There is more and more interests on distributed estimation in many practical systems in recent years. In previous chapter we studied some of algorithms for distributed density estimation in which the desired distribution is modeled using mixture of Gaussians. The parameters of this distribution are estimated by distributed and decentralized implementations of the Expectation Maximization algorithm.

In this chapter, our goal is to argue for estimating a global model using a decentralized solution for distributed data systems. We try to estimate a collective model which only requires moderate computation at each node and little data to transit between nodes. We propose an approach for aggregating the models in which both properties are obtained. In our approach, the aggregating of models is performed via their (few) parameters, rather than via multimedia data. Mixture models are in fact concatenated, then reduced to a suitable number of Gaussian components. A modification on Kullback divergence leads to an iterative scheme for estimating this aggregated model. We provide experimental results on a speaker recognition task with real data, in a gossip propagation setting.

In the remainder of this chapter, we first present an overview of the system in section 3.2, then we detail the proposed approach for mixture aggregation in section 3.3. In the section 3.4 we present an application of aggregation the GMMs. In section 3.5 a validation in the example of a speaker recognition task is provided and finally in section 3.6 there are some concluding remarks.

3.2 Overview of the system

In this section, we present an overview of the system that we propose for the distributed learning multimedia recognition to retrieve metadata in large and decentralized networks.

3.2.1 Nodes

For better understanding the problem and proposing the solution, here we describe the structure of a general node in a distributed system. Each nodes may have six partitions as follows:

1. user interface: a partition that interacts with user. It has ability to get a multimedia document from the user, preparing it as a query and sending the query to the network. The user interface is responsible to process the query answerers received from other nodes (usually as metadata) and return them back to the user. Another duty of user interface is to get a multimedia data and related metadata from user to store them in the databases of his node.
2. index partition: this part use a learning algorithm to index the multimedia and store the indexing parameters when user enter a multimedia data to be store in database.
3. retrieval part: it receives queries from other nodes and send the best answer to them. A node receiving a query, first evaluates the query against its own database and returns the results (or pointers to the results). After receiving query (audio, image,... or a feature vector), it will be recognized by recognition algorithm. The DBMS usually returns to the sender of query the pointers to the multimedia or metadata of the desired class that are stored in the its database.
4. training and learning partition: the nodes need a multimedia learning and recognition system for the two last partitions. Learning partition has duty to train the recognition system. But for having a good pattern recognition system, learning cannot be local, because it should learn increasingly and the system should train with many data. It can use the following ways for learning :

   (a) new multimedia documents that are given to index partition for training the system.

   (b) the data for training can be taken from the web pages and databases that exist on the Internet.

   (c) by getting the knowledge of other pattern recognition services: there are many multimedia recognition services on the network which can collaborate to do best recognition, (without relying on a central server). The nodes can exchange their learning and recognition knowledge. Some recognition systems on the network may use the same algorithm but have been trained with better examples. Nodes on the network can have better learning partition by getting the parameters of the other learners and combine them with their own learner.
5. database: in this part the format of data are stored, multimedia, metadata and the parameters of different classes of recognition system.

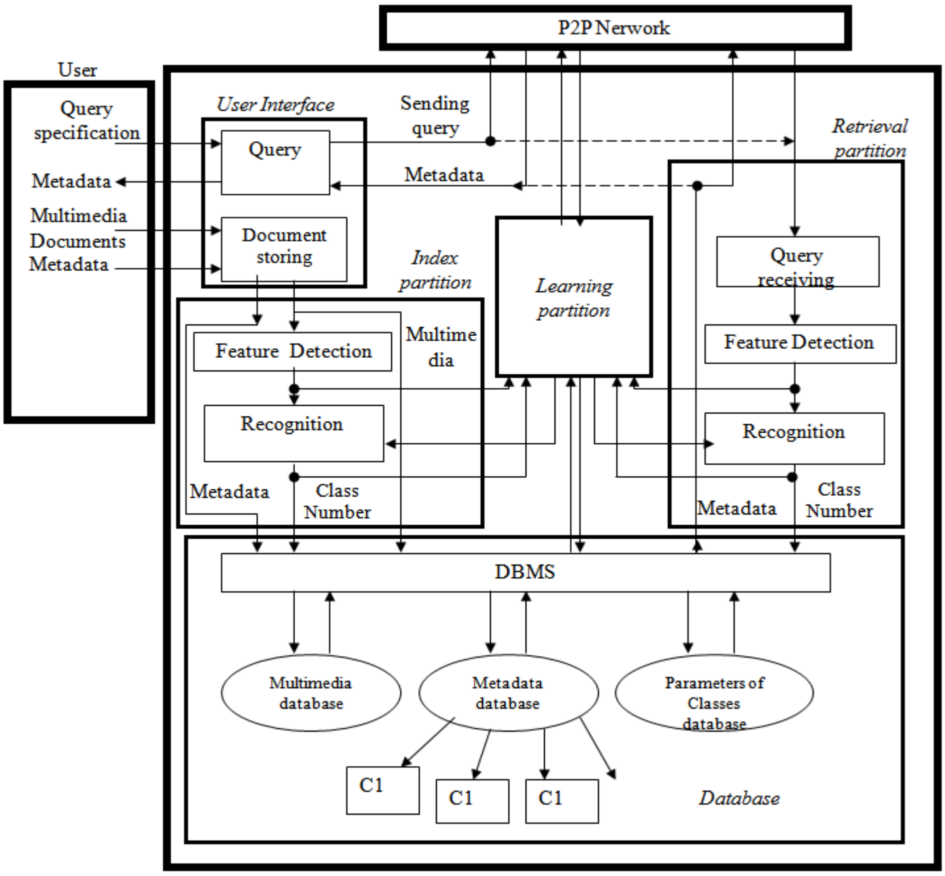


Figure 3.1 – A simple node on the system

6. services: each node may contain some services that share with other nodes, one of the services can be multimedia pattern recognition service that is our subject.

Figure 3.1 shows the partitions of a simple node.

3.2.2 Peer to peer network

We assumed a P2P distributed network for our problem because in this setting:

- the nodes can be client and server so each node may include a learning service. We consider nodes would run a service providing supervised learning of a multimedia class and would possibly store some training data.
- the nodes and services may join or leave the network at each time; the user can install a new service or uninstall a service on the node. So the structure of the network change

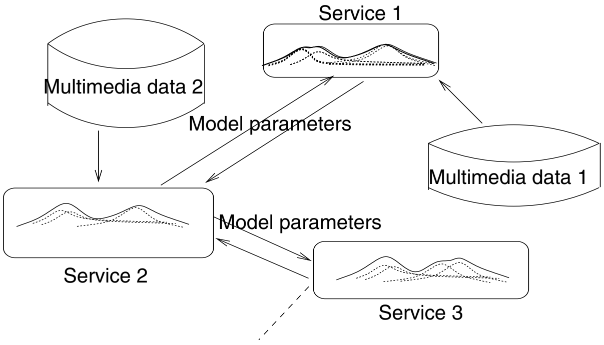


Figure 3.2 – Small-scale example of estimating a global model using mixture aggregation

continuously.

- the learning services can be aggregated to have a global learning service.
- the system is decentralized, so each node can supply data or learning tools, without any central administration.
- it is able to scale up to very large configurations.

3.2.3 Other characteristics

As we discussed in previous chapters we use:

- Gaussian mixture model which is a generative probability density estimation to estimate the models from data on the nodes.
- gossip-based computation to fusion the models.

3.2.4 Discussion

In a decentralized P2P network, the multimedia information is distributed on the whole network. We formulate the distributed multimedia learning recognition as a supervised learning problem for multimedia pattern recognition system on a peer-to-peer network. The idea of distributed learning multimedia recognition system is like a P2P multimedia files sharing applicant, but nodes offer :

a) pattern recognition application with a learning algorithm.
b) data with possibly metadata which feeds learning algorithms.

This system allows the pattern recognition share and collaborate with other pattern recognitions in the network. In this work, nodes in the network are willing to cooperate and share their knowledge and learn incrementally. Incremental algorithms take reliability into account which is necessary for having a good recognition system. In practice, this implies merging communicating this knowledge, merging it and dispatch it back.

Fig. 3.2 provides a small-scale example. S1, S2, S3 are three services doing learning and classification. S1 (resp. S2) estimates mixture parameters from data set 1 (resp. data set 2). Later, S1 and S2 discover one another and parameters from S1 are sent to S2 (or vice-versa). A common model is computed by combining the incoming mixtures and sent back to both S1 and S2. This resulting mixture can later be combined with a third mixture coming from S3, while always maintaining a suitable number of Gaussian components in the mixture.

The proposed the technique bears the two main following features :

- merging the estimated density on the nodes only involves transmission of, and computation on, mixture model parameters, rather than the generally large amount of multimedia data (or feature vectors that represent it). As a result:
  - the amount of information to be sent on the network is very low;
  - computation on nodes remains low.
- learning is conducting by gossiping and during the complexity of any mixture (i.e. the number of Gaussian components) remains related to the need.

A work which goal is close to ours, i.e. gossip-based distributed estimation of the parameters of a Gaussian mixture, has been presented in [99]. Their approach consists in introducing parallelism in the EM algorithm, by gossiping the M-step, resorting to original data. In our case, in contrast, each contributing node is in charge of estimating its local Gaussian mixture model, and is free to use any mixture model parameter estimation technique for this. The latter point gives an interesting degree of freedom towards a completely decentralized system : only the mixture description need to be standardized, while the node may benefit from recent advances in mixture estimation techniques (e.g. variational Bayes [8], or versions suitable for large amounts of data [190]). Further, the averaging in [99] between the parameters to be merged is simply uniform. To our understanding, a more central difference is that their way of merging knowledge between mixture models does not (at least explicitly) address correspondence between components to be merged, and leaves open the issue of merging models with different number of components. More generally, we shall see that our technique is amenable to variation of the number of components in the mixture along the gossiping process.

3.3 Distributed learning for multimedia pattern recognitions

As the classes are represented by a mixture of density, they can learn from each other by aggregating the models. This section details how mixture models may be merged using parameter-level rather than data-level computation. The first step is to fusion the mixture models and find a new model. The fusion of the two GMMs produce one GMM, that has $m_1 + m_2$ components which m_1 and m_2 are the numbers of Gaussians in two model. But, it is not efficient to have sum of the densities of two or more models. We should then reduce the sum of the densities that has minimum distance to the two model under some distance measure. This lead us to an optimization problem. In this section we talk about the aggregation of Gaussian Mixture models and the optimization algorithm for it.

3.3.1 Aggregation of the learning systems

Let two nodes each carry different probabilistic Gaussian mixture models, denoted $M_1(x)$ and $M_2(x)$, associated to the same multimedia entity and hence hidden density $p(x)$. The mixtures can be expressed as :

$$M_k(x) = \sum_{i=1}^{m_k} w_k^i N_k^i(x), \qquad k = 1, 2 \tag{3.1}$$

where $N_k^i(x)$ is a Gaussian component which mean is μ_k^i and covariance Σ_k^i and the w_k^i are scalar weights. Model M_k is estimated on a data set of size n_k located on node k. $p(x)$ can be estimated by concatenating incoming mixtures as follows :

$$M_c(x) = \frac{1}{n_1 + n_2}(n_1 \sum_{i=1}^{m_1} w_1^i N_1^i(x) + n_2 \sum_{i=1}^{m_2} w_2^i N_2^i(x)) \tag{3.2}$$

However, the $m_1 + m_2$ components in M_c are generally largely redundant, which implies a useless increase in evaluation cost of likelihoods for this density at query time, when merges are chained by gossip. Consequently, scaling up the scheme requires transforming M_c into a reduced mixture $M_r = \sum_{i=1}^{m_r} N_r^i(x)$ that preserves reasonably well the density while only having the necessary number of components for this. The point of this policy is that the order of magnitude of the number of components is kept constant through propagation, although in detail it may fluctuate to fit the complexity of the density.

3.3.2 Definition and optimization of the similarity between M_c and M_r

We want to cluster the components of M_c into a reduced mixture of $m_r < m_c$ components. If we denote the set of all Gaussian mixture models with at most m_r components by $\mathcal{M}_{m_r}$ one way to formalize the goal of clustering is to say that we wish to find the element M_r of $\mathcal{M}_{m_r}$ "closest" to M_c under some distance measure. The class models in the nodes would be used to classify new data, typically based on maximum likelihood or more elaborate criteria involving the likelihood. In order to preserve the likelihood as much as possible, we seek a mixture model M_r which maximizes the expected log-likelihood of data D assumed to be drawn from M_c, see (3.3). It is classically established [18] that this amounts to minimizing the Kullback-Leibler divergence $KL(M_c \| M_r)$, defined by (3.5), which, in short, measures the loss of information due to the approximation of M_c by M_r:

$$\begin{aligned} \hat{M_r} &= \arg\max \quad \mathbb{E}_{M_c} \; [\; ln \;\; p(D|M_r) \;] && (3.3) \\ \hat{M_r} &= \arg\min \quad \left[-\int M_c(x) \; ln \; M_r(x) \; dx \right] && (3.4) \\ \hat{M_r} &= \arg\min \quad \left[-\int M_c(x) \; ln \; \frac{M_r(x)}{M_c(x)} \; dx \right] && (3.5) \end{aligned}$$

A major issue for the practical computation of (3.5) is the lack of closed form for this divergence, in the case of Gaussian mixtures. We propose a bypass in the form of the following approximation.

Linearity of integral applied to (3.4) provides:

$$\hat{M_r} = \arg\min \left[-\sum_i w_c^i \int N_c^i(x) \ln M_r(x) \, dx \right] \tag{3.6}$$

In each term of the sum in equation 3.6, we approximate the mixture M_r by only one of its Gaussian components, selected as the best approximation to N_c^i in the KL sense. This leads to the following similarity measure:

$$d(M_c, M_r) = \sum_{i=1}^{m_1+m_2} w_c^i \min_{j=1}^{m_r} KL(N_c^i \| N_r^j) \tag{3.7}$$

where N_c^i (resp. N_r^i) is the i^{th} component of M_c (resp. of M_r). This similarity measure exhibits two good properties :

- it can easily be computed at low-cost, since the Kullback divergence between two Gaussians, which parameters are (μ_1, Σ_1) and (μ_2, Σ_2), benefits from the following closed-form expression :

$$\frac{1}{2}(log\frac{|\Sigma_2|}{|\Sigma_1|} + Tr(\Sigma_2^{-1}\Sigma_1) + (\mu_1 - \mu_2)^T \Sigma_2^{-1}(\mu_1 - \mu_2) - \delta) \tag{3.8}$$

  where δ is the dimension of the feature space.

- it preserves the following theoretical benefit of the original Kullback divergence : as the amount of data grows, minimizing $d(M_c, M_r)$ is equivalent to maximizing the expectation of the data log-likelihood (data being drawn from M_c), also assuming that all data drawn from a component of M_c is assigned, in the mixture reduction process, to the same component in M_r.

3.3.3 Optimization : an iterative scheme and its initialization

We optimize locally criterion 3.7 with an iterative scheme detailed in algorithm 6, which is adapted (by several aspects) from a technique [70] proposed in the context of hierarchical clustering. The optimal reduced model $\hat{M_r}$ is the solution to the minimization of equation 3.7 over $\mathcal{M}_{m_r}$ (the space of all mixture wit m_r components):

$$\hat{M_r} = \arg\min_{M_r} d(M_c, M_r) \tag{3.9}$$

Although the minimization ranges over all the $\mathcal{M}_{m_r}$, the optimal density $\hat{M_r}$ is a model obtain from grouping the components of M_c into clusters and collapsing all Gaussian within a cluster into single Gaussian.

The procedure bears analogy with the classical k-means algorithm, in that it operates local optimization by alternatively assigning elements to groups and re-computing group representatives. In our context, the elements are the components of M_c and the representatives those of M_r. To gain insight into complexity, we assume $m = m_1 \approx m_2 \approx m_r$. The search space is of size $O(m^2)$ and typically cannot be searched exhaustively if there are more than 10 components which is common when modelling multimedia classes.

The iterative technique proposed assumed some initialization for assigning the components of M_c onto those of M_r. If no prior knowledge is available, as often done with k-means, the initial assignments π^0 from which local optimization proceeds could be drawn randomly. Our context suggests a more effective initialization criteria : since generally, Gaussian components coming from the same mixture are not redundant, we draw π^0 at random with the constraint that components arising from the same mixture are not initially grouped. The iterative scheme may still regroup them later, if the data drives it that way. As we draw multiple starting points to retain the best local optimum, this strategy improves sampling of the search space.

Algorithm 6 Iterative optimization algorithm for estimating the reduced model M_r (criterion (3.7))

```
for $m_r$ : from 1 to m1+m2 do
  Start from a random initialization $\hat{\pi}^0$ (or given, if available)
  $it = 0$
  repeat
    1. Re-fit mixture $M_r$ :
    given the current component clustering $\hat{\pi}^{it}$, set initially or computed at the previous iteration, update mixture model parameters as follows :

        $\hat{M_r}^{it} = \arg \min_{M_r \in \mathcal{M}_{m_r}} d(M_c, M_r, \hat{\pi}^{it})$    (3.10)

    where $\mathcal{M}_{m_r}$ is the space of all mixture with $m_r$ components that may be formed by grouping components of $M_c$. This re-estimation in fact amounts to updating each component of $M_r$ as follows. For component $j$, algebra leads to the following expressions :

        $\hat{w}_r^j = \sum_{i \in \pi^{-1}(j)} w_c^i, \qquad \hat{\mu}_r^j = \frac{\sum_{i \in \pi^{-1}(j)} w_c^i \mu_c^i}{\hat{w}_r^j}, \qquad , \hat{\Sigma}_r^j = \frac{\sum_{i \in \pi^{-1}(j)} w_c^i (\Sigma_c^i + (\mu_c^i - \hat{\mu}_r^j)(\mu_c^i - \hat{\mu}_r^j)^T)}{\hat{w}_r^j}$    (3.11)

    where $\pi^{-1}(j)$ is a light notation for $\hat{\pi}^{-1,it}(j)$, the set of $M_c$ that project onto component $j$ in $M_r$. Let us note that $\hat{\Sigma}_r^j$ is generally non-diagonal, even if the components being grouped have diagonal covariance matrices, such as is often the case with decorrelated features used in e.g. speech or speaker recognition.
    2. Grouping components :
    for mixture $\hat{M}_r^{it}$ obtained in Step 1, we seek the mapping $\pi^{it+1}$, defined from $\{1, \ldots, m_1 + m_2\}$ into $\{1, \ldots, m_r\}$, which best groups components of $M_c$ to build components of $\hat{M}_r^{it}$, in the following sense :

        $\hat{\pi}^{it+1} = \arg \min_{\pi} d(M_c, \hat{M}_r, \pi)$    (3.12)

    In other words, each component $i$ of $M_c$ projects onto the closest component $j$ of $\hat{M}_r^{it}$, according to their Kullback divergence ((3.13) below). In this phase, we resort to exhaustive search among 'source' components, which has a low-cost, thanks to the availability of (3.8).

        $\pi^{it+1}(i) = \arg \min_{j} KL(N_c^i || N_r^j)$    (3.13)

    3.it=it+1
  until convergence (i.e. $\pi^{it+1} = \pi^{it}$)
  compute $d(M_c, \hat{M}_r) = \sum_{i=1}^{m_1+m_2} w_i \min_{j=1}^{m_r} KL(N_c^i \| N_r^j) \quad + \quad \nu_{M_r}$
end for
Retain model $\hat{M}_r$ which minimizes $d(M_c, \hat{M}_r)$ over the set of candidate mixture complexities explored.
```

3.3.4 Complexity of reduced model

An important point in the proposed approach is the determination of the number of Gaussian components in the reduced model M_r. The seminal study reported in [3] showed that estimating the Kullback divergence is in fact affected by a bias that grows with the number of parameters to be estimated, i.e. with the number of components. It also supplies a first-order approximation of this correction, which we apply here to the definition of $d(M_c, M_r)$, which hence becomes :

$$d(M_c, M_r) = \sum_{i=1}^{m_1+m_2} w_c^i \min_{j=1}^{m_r} KL(N_c^i \| N_r^j) \quad + \quad \nu_{M_r} \tag{3.14}$$

where ν_{M_r} is the number of independent parameters in the mixture. Our experimental results (not reported here) back the application of this approximation : the number of components obtained in practice appears very similar to that obtained by usual (AIC,BIC) model selection criteria on the model computed directly on all the data (i.e. discarding the distributed aspect of the learning process). We evaluate exhaustively from 1 to $m_1 + m_2$ the performance of each possible number of components in M_r, in independent trials. A faster alternative would be to compute this recursively downwards from $m_1 + m_2$ to 1, but experimental results suggest this can excessively prune the search space at early stages.

3.4 Application to tracking a hierarchy of partitions for geotemporal organization of personal images

In this section, we present one of the applications of aggregation of mixtures for automatically organizing large personal image collections in camera-equipped mobile devices. We propose a technique based on a two-level hierarchy of mixture models to automatically organize such a collection with the geo-temporal meta-data attached to each pictures

3.4.1 The goal and existing work

Building of personal image collection from mobile devices is now widespread, due to the quality improvement of integrated photographic sensor and users habits evolution face to new services as MMS (Multimedia Messaging Service). Composed of several thousand images, such a collection involves the need of new tools to efficiently retrieve/browse its content. This area of research is of much interest and presents several answers from industry and academics, as NokiaLifeBlog, Microsoft MyLifeBits and PhotoCompas. The particularity of the task lies in the availability of new meta-data provided by the acquisition device (time, location and camera settings) and the favorite querying/browsing user's criteria. Indeed, users studies (e.g. [155]) conclude that social interaction, events, time and places are the favorite criteria to organize such a collection. In this field of consumer images, some work has addressed image content-based supervised classification. In contrast, we focus, in this section, on the sole use of temporal and geolocation meta-data attached to each picture. We assume location are coordinates provided by a GPS/E-OTD type of equipment, i.e. the data is a stream of $\{(t, (x, y)) \in R \times R^2\}$ elements. The contribution put forward is a technique (statistical criteria and algorithms for optimizing them in an incremental manner) for automatically building a two-level hierarchical organization of images, based on their time and geolocation stamp. Distinct hierarchical classifications are built for time and space, respectively from temporal and geolocation data, but are tackled with almost identical techniques. The hierarchies of partitions are built incrementally, as data flows in, since the image acquisition and browsing phases are highly interleaved. This work is designed with browsing, rather than querying, in mind, especially over a mobile device with limited man-machine interaction. Several approaches were proposed to segment an image collection. The work closest to ours [127] organizes the data set based on time and location clusters. Their strength is on handling both cues jointly, but its implies more arbitrary parametrization that our approach and is not incremental.

In our view, incrementality is a requirement to keep the collection organized without user needing to think about it. Running on a mobile phone as a permanent background task with low priority, the computational demand of our technique is then far less than, for instance, real-time video codecs currently running on such platforms. Overall, the main problems to resolve here are to cope with particular data stream (time-stamp or location), accommodating various shapes of groups without excessively arbitrary parametrization, and to provide a hierarchical representation, maintained over time.

3.4.2 Two-level Partition Tracking

When grouping images to ease browsing among them on a small-size display, both a small number of groups, driven by the screen size, and a number of groups driven by the sole data, are sensible criteria. We assign each of these two options to respectively the coarse and the fine level of the hierarchy. The recovery of the hierarchical geo-temporal structure consists in building and updating, in an unsupervised manner and as new data flows in, the four partitions of the data set (two geographic and two temporal). We outline here the main features of the proposal.

1. fine partitioning mainly follows the technique described in [141]. It is based on probabilistic Gaussian mixture modeling of the data. The original integrated completed likelihood optimality criterion provides robustness against non-Gaussianity; a semi-local search strategy with splits & merges improves resilience to local minima w.r.t. standard EM; a Bayesian estimation of the covariance matrices help preserve reliability even for clusters that are assigned small amounts of data.

The choice of mixture models grants two advantages :

- it breaks the combinatorial explosion inherent to data grouping problems,
- it suits well the incremental nature of the task, since data-to-class assignments may evolve in a flexible way as new data streams in, using a light predict/update mechanism.

2. coarse partition of the data is determined by identifying suitable groups of the fine partitions (the technique differs completely from the work in [141]). Whereas the number of clusters in the fine partition is fully determined from the data (Bayesian implementation of Occam's razor), the coarse partition has a low number of clusters, in order to provide an overview of the collection for browsing and possibly zooming into the fine partition. Grouping of fine-scale mixture components is formulated as the identification of a coarse mixture model that minimizes a modified Kullback-Leibler loss in approximating the fine by the coarse model. The combinatorial problem of grouping fine-scale components is solved by an iterative algorithm, adapted from [70], that operates by alternatively estimating parameters of the coarse model and updating fine-to-coarse component assignments. It resembles the fine-scale data clustering technique, but it operates on fine-scale Gaussian components rather than on the initial data. The contribution

of this work are mostly due to the conjunction of the following properties of the dual partition :

- it has a well-founded definition of the loss of information between the fine and the coarse partition,
- the computation of the coarse partition only implies a very low cost, since it only resorts to the fine-scale model parameters rather than the data,
- it is amenable to a predict/update mechanism as new data flows in, by initializing the iterative grouping technique at the configuration previously obtained at convergence. Besides reducing computations, this ensures also stability of the coarse partition over time, which is important for the user experience ; these properties are not enjoyed by of classical techniques such as Agglomerative Hierarchical Clustering,
- it is not affected by non-Gaussianity of the fine-scale clusters, which is the case of approaches that try to match directly a hierarchy of mixture models on the data.

3.5 Experimental results

The example of distributed speaker recognition is taken throughout this section, as it is a representative case where Gaussian mixtures are very popular. The technique however directly applies to a wide range of audiovisual classes. The experimental results obtained on real data. We first focus on the merging operation, i.e. at local scale (section 3.5.1). We then observe global performance, in the context of gossip-based mixture propagation (section 3.5.2). Throughout these experimental results, the figure of merit is the quality of the class-conditional pdf estimate, in particular with respect to a conventional, centralized approach, rather than ability of the scheme to classify new data correctly. The latter however derives directly from the former in a Bayesian decision rule. There is also experimental result (section 3.5.3) for the application discussed in previous section to tracking a hierarchy of partitions for personal images.

3.5.1 Detailed view on one or two merge operation

In the first experiment, three nodes each have learnt a probability density for speaker 'A' in a common 13-dimension mel-cepstral feature space[148]. Three corresponding mixtures are merged simultaneously into a single mixture (i.e. equation (3.1) generalizes to merging more than two mixtures). Each node was provided with different training data from the same speaker and the duration of audio recordings was between 7 to 16 seconds. i.e. rather short for training. Each node provides a mixture estimate from local data, and is free to choose the precise technique used for this.

For our experiments, the Expectation-Maximization local optimization algorithm is employed, but with some enhancement [16] to limit local minima and ensure the three mixtures are reliable inputs. Each mixture also autonomously and automatically determines its number of components (in practise, using the common BIC criterion). All covariance matrices in the mixture are full (rather than spherical or diagonal). Let us point out that this first phase only provides the mixtures to the main contribution of the work, they could be generated in another way.

The three incoming nodes respectively have 4,4 and 5 components. Their concatenation into M_c supplies a 13-component model, which should be reduced to a number of components to be determined. Fig. 3.3(a) displays, in the representative example case of the second feature vector, the three incoming densities, the concatenated density and the density after mixture reduction. The mixture estimated (again enhanced EM) on the whole data over network is also plotted. While the main point of this thesis is to propose a decentralized alternative to this, the direct model serves herein a reference density against which we evaluate the loss due to distribution of the data and computations. Fig. 3.3(b) shows that criterion (3.14) chooses a reduction from 13 to 4 components. To evaluate the effectiveness of the mixture reduction, fig. 3.3(c) provides numerical evidence in terms of Kullback loss between reference mixture and approximating mixtures. KL divergence is used (rather than its approximation proposed in equation 3.7). We evaluate Kullback divergence by a Monte-Carlo procedure with N=10^8 samples, as follows :

$$KL(p, \tilde{p}) = \frac{1}{N} \sum_{i=1}^{N} log \frac{p(x)}{\tilde{p}(x)} \tag{3.15}$$

where p and $\tilde{p}$ respectively denote an ideal model and its approximation. While this should be closer to the true loss than (3.7), its computational cost forbids its usage in the scheme, it is only used here for external assessment.

It can be observed that the direct mixture is much better approximated by the reduced mixture than by any of the incoming mixtures. This does not come at the expense of mixture complexity, since the reduced mixture has 4 components, in fact the same is estimated by a BIC criterion for the direct model.

We report a second experiment, applied to a different speaker. It again involves three nodes but, similarly to fig. 3.2 and in contrast to the previous experiment, two nodes are merged, and then a third node is merged to their reduced mixture to form a final reduced mixture (illustrated in figure 3.4) . The experiment is conducted in a 2-dimension space (second and third cepstral feature vectors), for the sake of clarity of fig. 3.5(a), which shows all original feature vectors (i.e. the training data). Its purpose is more an illustration value than a demonstration of large scale effectiveness. The mixtures have respectively with 3, 5 and 4 components. The centers of the incoming mixtures, as well as the centers of the concatenated (in second step), reduced (in second step) and direct mixtures are superimposed to the data, and the two latter are clearly very close. (Note that the mixture model 3 entered in second step for aggregating and all components of this model are part of concatenate model in second step, so its component centers are the same as some of the components in concatenated model in the figure 3.5(a)). Figure 3.5(b) shows the Kullback divergence of the difference between reference mixtures and approximating mixtures which shows again the final reduced model can better represent the direct model comparing to the initial mixtures.

3.5.2 Application to combination and exchange by gossip information spreading

To evaluate the performance of the proposed technique in the distributed context described in the introduction of the section, a mechanism is required for propagating mixture representations from the edge of the network (nodes that own data) to all nodes. We employ here a simple gossip (or epidemic) spreading procedure, described in Algorithm 7. Despite its simplicity, it enables fast and robust propagation in an asynchronous, decentralized manner that fits well the peer-to-peer viewpoint. Our example networks are fully connected and hence each node owns data, but this is by no means a requirement.

Algorithm 7 A gossip cycle for merging-sharing Gaussian mixture models

```
1. Select at random two nodes in the network, which models are $M_i$ et $M_j$ (practically, nodes should autonomously select their partners in a dialogue)
2. Concatenate $M_i$ and $M_j$ into a single model $M_c$, then reduce this to $M_r$
3. Assign $M_r$ to $M_i$ and to $M_j$
```

A network of 13 nodes is used in the first experiment. Each node owns different data from the same speaker and independently estimates its own model. Practically EM with multiple starts is employed in our experiments for this purpose but, as stated before, other techniques may be used.

To evaluate the capability of mixtures on the network to model data D from the class of interest (here, a speaker), the classical marginal likelihood is selected [114]. Data D here is the union of the data from the same speaker dispatched over all nodes, which is never gathered when the practical system runs, but is a relevant figure of merit for external observation . The practical computation of the marginal likelihood of the data is carried out with the BIC criterion :

$$BIC(D|M) = -2\log p(D|\hat{\theta}) + \nu \log(\sharp D) \tag{3.16}$$

where mixture M is defined by a parameter vector θ, $p(D|\hat{\theta})$ is the likelihood of the data for this model, ν is the number of independent parameters in the mixture and $\sharp D$ the size of the data set (the data set does not need to propagate in the network, but its size should propagate and cumulate in n_1 and n_2)

Figure 3.6(a) depicts, after each gossip cycle and on each node, the evolution of criterion (3.16), which should be minimized. The following observations can be made :

1. the process stabilizes around a ”collective model”. Convergence cannot be established, as illustrated in the zoom fig. 3.7, due to the lack of an optimization criterion global to the network, which is the case in the prototypal example of computation of a mean [99, 22]. From a practical viewpoint, however, all nodes are rapidly assigned a mixture that is better (slightly or largely) than any of the original mixture, which later implies improvement in recognition rates when the system is queried. This information is summarized in figure 3.6(b), which plots the mean and variance of the BIC criterion, over the set of nodes.
2. the effectiveness of the collective model is significantly better than that of a single mixture model that could have been estimated directly on the whole data (the performance of which is represented by a dashed horizonal line). This latter advantage however reduces when the size of the feature space is large compared to the amount of training data (dimensionality curse). Overall, however, this example, which is representative of many other obtained, suggests that the proposed scheme provides promising results on three points : quality in model estimation, the flexibility of a decentralized system, and speed up thanks to parallel computing.

It should also be underlined that the horizontal axis only indicates order and is non-linearly related to time, since gossiping is strongly parallel.

As illustrated in fig. 3.8, the scheme can easily handle a node that joins the network. In this example involving 20 nodes, a additional node joins after 50 cycles. Soon after it joins, it benefits from the previous exchanges. Indeed, the amount of data available for mixture estimation ((n_1 and n_2) in eq. (3.2) cumulates as gossip progresses.

3.5.3 Experiments for the application of tracking a hierarchy of partitions for personal images

To evaluate the proposition of tracking a hierarchy of partitions for personal images which is discussed in section 3.4 , an experiments is carried out on a real personal image collection, involving 721 images taken over three years in several countries. The number of groups in the compact partition is set to a maximum of 4 (but if the analysis finds that less better describes the data, it will conclude with less) Fig 3.9 presents the geolocation partition obtained. The fine partition ('+') includes very compact classes, since the data naturally forms highly compact clusters. The user generally took many pictures in precise locations; all 25 of them are reasonably retrieved. The coarse partition (dashed line) contains three classes (rather than four from the initialization of the iterative grouping), as this is rated more desirable. The present work is founded on the belief, backed by user studies, that it is useful to group personal pictures according to time and space. We proposed a technique addressing this goal through two-level tracking at low cost, statistical criteria and optimization techniques, that satisfy several specific requirements of this task, which is not the case of conventional clustering techniques. Examples of open issues include (1) the labeling of groups in the most informative way, which is problematic in that data-driven time and space groups do not directly correspond to simple temporal or geographic labels from a GIS (Geographical Information System), (2) How to make good, joint, use of temporal and spatial meta-data.

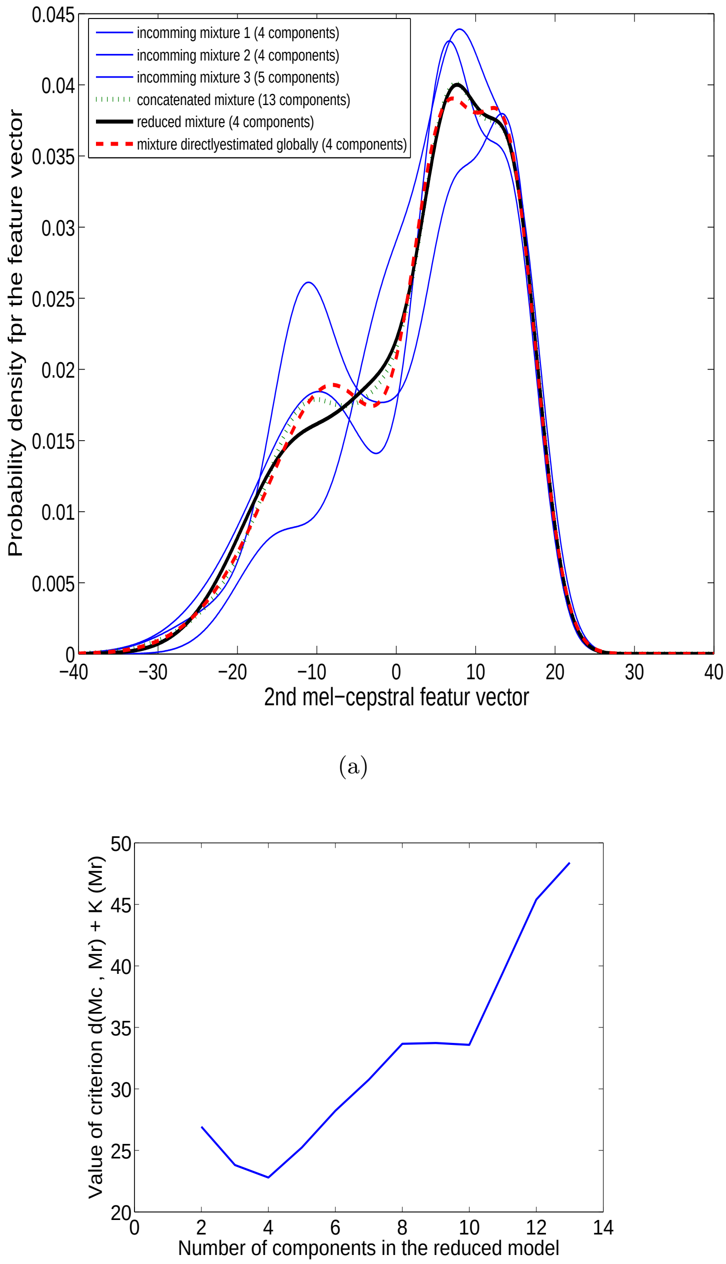
3.6 Conclusion

The work which we described in this chapter fits into a vision towards a multimedia indexing and retrieval system, which would be decentralized and deployed on a large scale. In this system, the pattern recognition services in the nodes on the network can share their parameter so they can learn increasingly from each other by combining their parameters. In this setting, algorithmic components are required, that induce low computational cost, incrementally and only require a little amount of information to transit between nodes. We proposed a novel scheme for this purpose, dedicated to Gaussian mixtures models, which are one of the most useful representations of a multimedia class. The proposal wraps a parsimonious mixture model merging technique into a gossip framework, demonstrating that it can efficiently propagate and collectively improve estimates over time. The point of the gossip framework is that it is well suited to dynamic, decentralized computing environments. Efficiency of the proposed technique comes the two main following features:

- merging density estimates between nodes only involves transmission of, and computation on, mixture model parameters, rather than the generally large amount of multimedia data (or feature vectors that represent it). As a result:
  - the amount of information to be sent on the network is very low ;
  - computation on nodes remains low, relatively to estimation tasks that operate on the multimedia data or feature vectors.
- during the gossip-based model learning phase, the complexity of any mixture (i.e. the number of Gaussian components) keeps a constant order of magnitude. We underline that the distributed learning phase and the querying phase, can fully overlap, since mixture reduction keeps the class representation directly ready for query evaluation.

Experimental results are encouraging in the sense that they confirm that the proposal achieves a good trade-off between the quality of the mixture learned and disseminated in this collective manner, in terms of its ability to represent the density of the data and hence classify forecoming data and, on the other side, the computational cost, which is kept very low both at learning time and at querying time, since the number of components is kept moderate. However the major problem is the efficient algorithm to combine the parameters of different pattern recognitions that exist on the network.

More generally, crossing pattern recognition and large-scale distributed computing is a promising direction, since the first ingredient can greatly enhance services offered to users, far beyond file sharing, while the second provides data, computation and algorithmic resources.



(a)

(b)

	KL(concatenated ‖...)	KL(direct‖...)
mixture 1	0.0427	0.0430
mixture 2	0.0075	0.0071
mixture 3	0.0175	0.0182
reduced mixture	0.0018	0.0021

(c)

Figure 3.3 – (a)For the second mel-cepstral feature vector, density of incoming mixtures, of their concatenation and of the resulting reduced mixture. As a reference, the mixture directly estimated of the whole data is plotted. (b) Selection of the number of components : criterion 3.14 is evaluated for each candidate number of components and the optimal (lowest) value indicates the selected model (c) Kullback divergence of the difference between reference mixtures.

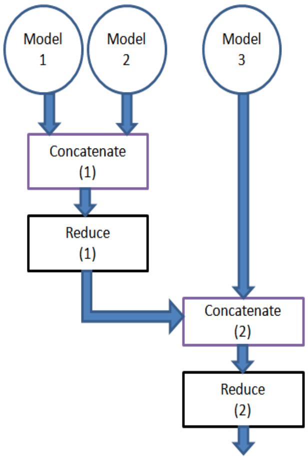
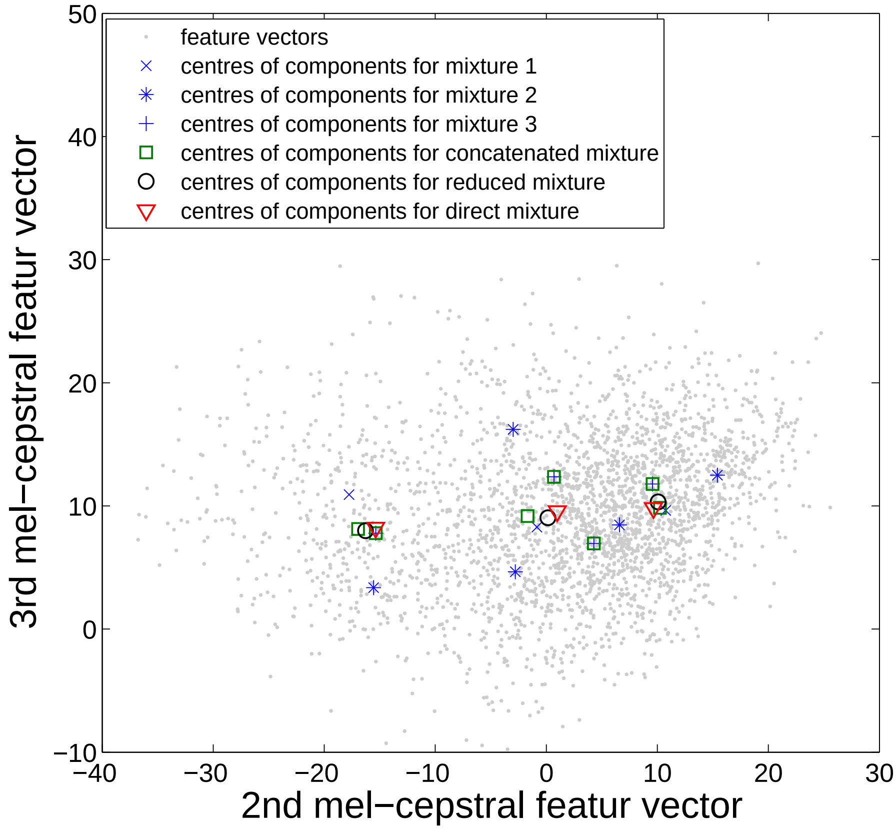


Figure 3.4 – Schematic of second experiment: two node are aggregated the third model aggregate with their result

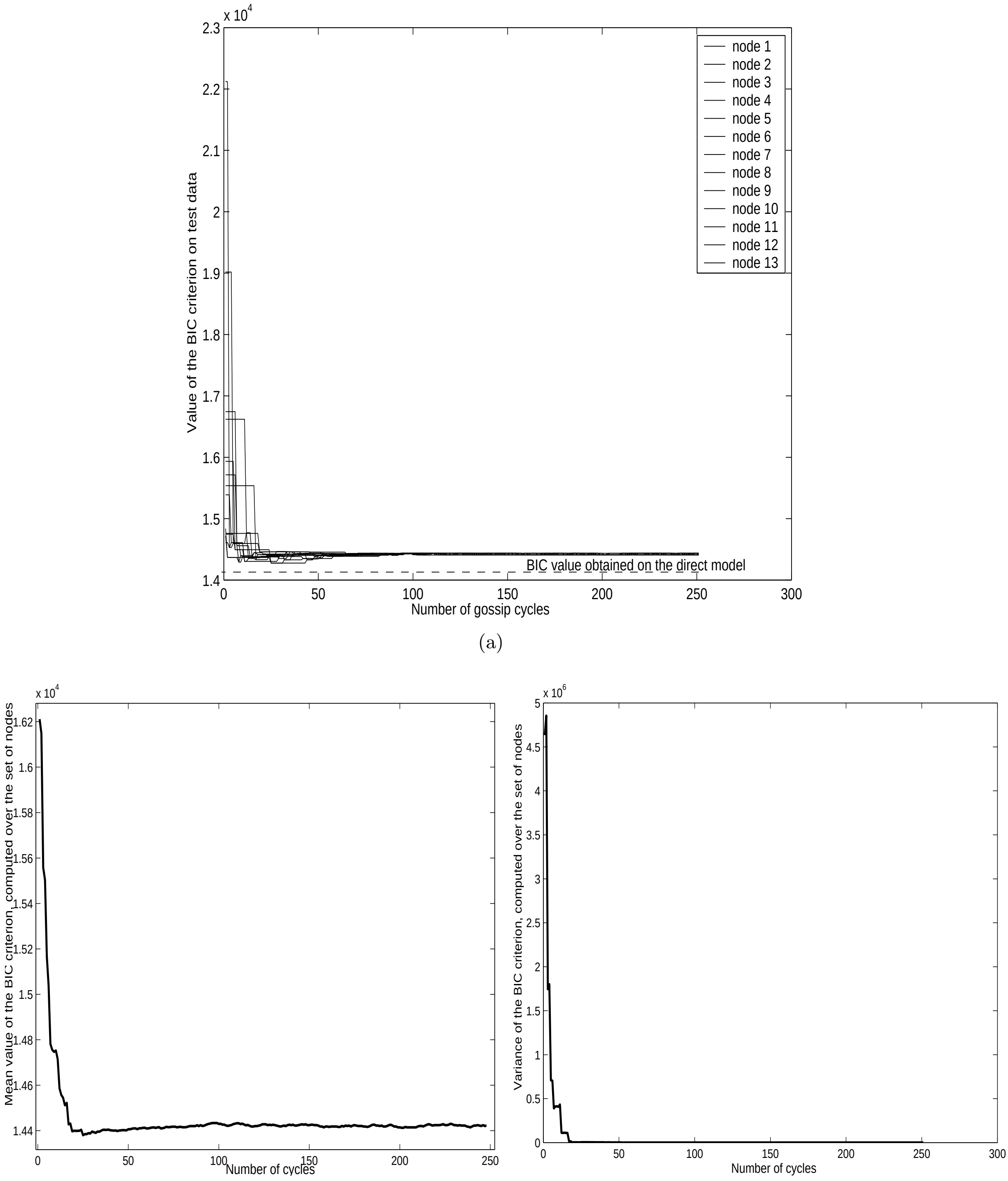


(a)

	KL(concatenated ‖...)	KL(direct‖...)
mixture 1	0.0510	0.0385
mixture 2	0.0957	0.0872
mixture 3	0.0725	0.0415
reduced model	0.0078	0.0223

(b)

Figure 3.5 – Three mixtures merge (not simultaneously). (a) shows the feature vectors and the centers of the Gaussian components (for incoming, concatenated (second step), reduced (second step) models, as well as, for reference, the mixture that could be directly estimated over the whole data set) (b) Kullback divergence of the difference between reference mixtures.



(a)

(b)

Figure 3.6 – In a 13 node network, evolution of the BIC criterion for each nodes is plotted as gossiping progresses. (a). A dashed line indicates the performance of the model estimated directly on the whole data set. (b)Mean and variance of the BIC criterion, over the set of nodes, are plotted as gossip progresses.

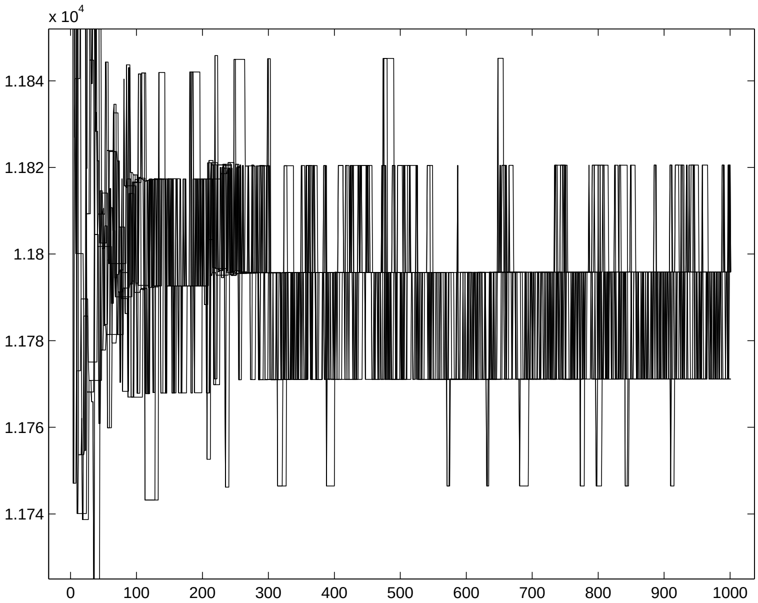


Figure 3.7 – In the same type of experiment as previously, the temporal evolution of the BIC criterion at each node evaluated on a test set shows not to converge, due to the lack of optimization global to the network.

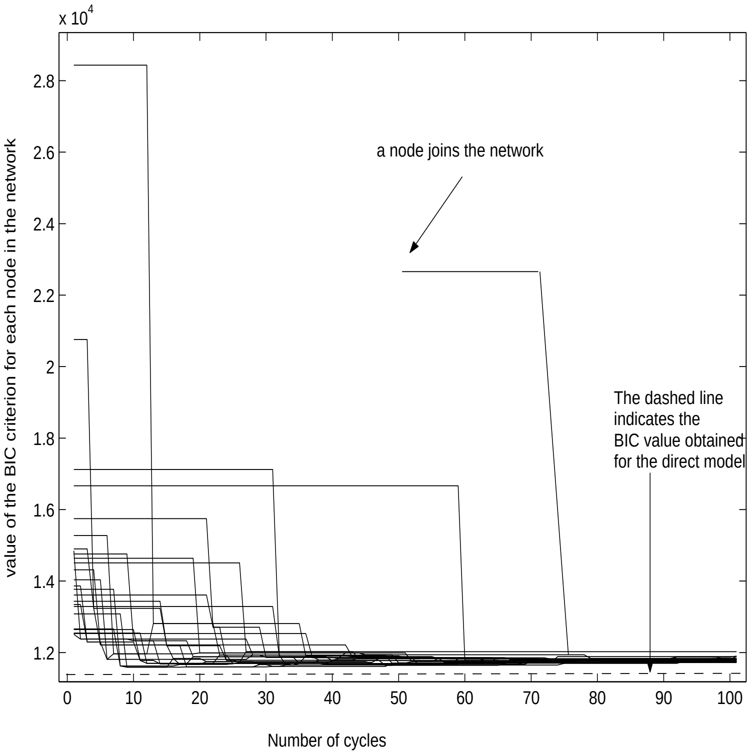


Figure 3.8 – This experiment illustrates fast integration of a node joining a distributed learning process involving 20 nodes. Right from its first contact (cycle 73), the joining node strongly improves by catching the central trend of the network.

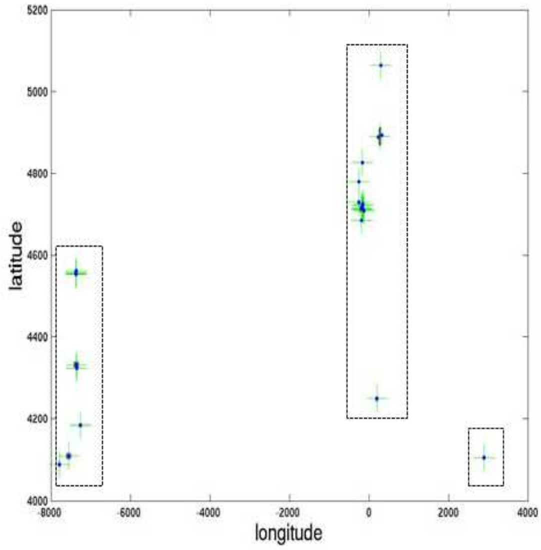


Figure 3.9 – Geolocation-based hierarchy. The fine partition is distinguished as the sign '+', while the coarse partition is displayed as dashed lines.

CHAPTER 4

Choosing the best models by cross-validation method

In the previous chapter, we discussed finding better model using merging the parameters of models in a gossip manner. In this chapter, we present an algorithm which uses a combination of cross validation and merging to find the best model among many models which exist on the network for a single class .

4.1 Introduction

In chapter 3 a low cost distributed learning for multimedia indexing using Gaussian Mixture Model has been proposed. In that approach we discussed the aggregation of probabilistic Gaussian mixture models of the same class but estimated on several nodes on different data sets. Mixture models are in fact concatenated, then reduced to a suitable number of Gaussian components. A modification on Kullback divergence leads to an iterative scheme for estimating this aggregated model. The approach for fusion only requires moderate computation at each node and little data to transit between nodes because aggregating applies to their (few) parameters, rather than multimedia data. Although this idea sounds interesting several difficulties remain unresolved. One is how to find the best models of the classes on the network. Also the convergence is not proved and it does not guarantee that merging two mixtures has better likelihood to the whole data than the two initial mixtures. In this chapter, we show an algorithm which uses a combination of cross validation and merging to find the best model among many models which exist on the network for a unique class. The algorithm has two parts :

- finding the best model of a class among many models for that class that exist on a region, by a cross validation manner.
- in order to create better models, we try to fusion the models by concatenating the models and reducing a proper number of components. By fusion we can find the new models which have higher likelihood with the same or lower complexity.

4.2 Finding best model using cross validation

As we said in chapter 3, we suppose that we have a P2P network and we focus on the learning of a probability density in a feature space common to all contributors in the network, for the class to be learned. Each node contains a Gaussian mixture model for each class. Each peer is responsible to estimate a Gaussian Mixture Model for each class, from its multimedia data. Our goal is statistical supervised learning of a unique class.

In chapter 3, this goal is tackled through an approach requiring only moderate computation at each node and little data to transit between nodes. Mixture models are in fact concatenated, then reduced to a suitable number of Gaussian components. Although this algorithm can decrease the BIC value of the most of GMMs which contribute in the process, but sometimes it increases the BIC value after aggregating some of the models. Also after aggregating the models, sometimes the likelihood to whole data is not better. In the other words, merging the model parameters does not always increase the model efficiency of the two mixtures that contribute in the merging. This is probably due to the supposition made in that algorithm according to which the node with larger number of data is a better model so it has more effect on the results. But it can not differ between the good and bad models before merging the models. In fact for some of the nodes, the model parameters may have not been estimated well and it can affect on the other mixtures when merging is occurred. So we need to distinguish the good models and the bad models, to prevent bad models affect the result when we merge the models. In other words, we need to find the model which has the best likelihood to the whole data, but as the network is large and multimedia data are huge, it is not simple to do that for whole the network. So we can find the best locally node on each region of the network. We consider each node and its neighborhoods as a region and use the cross-validation manner to find the best model in each region. This is done by assigning a score to each model that is calculated in cross-validation manner. Next we can compare the models and find best model according to the other nodes in this region. We first discuss about the cross-validation algorithm. Then we present our algorithm for finding the best models in a cross-validation manner.

4.2.1 Cross-validation

A method for evaluation the models is cross validation that sometimes called rotation estimation [98]. The basic idea in this method is that we don't use the entire data set when training a learner and before training, some of data is removed. The removed data can be supposed as new data and used to test the performance of the learned model. A whole class of model evaluation methods which use this idea are called cross validation. Comparing with residuals, the models can be better evaluated as in residual methods there is no indication of how well the learner will do when it see new data that has not already seen. In other words, cross-validation is the statistical practice of partitioning a sample of data into subsets in which a single subset (called the training set) is first used for analysis and then the other subset or subsets (called validation or tests sets) are used for validating the initial analysis.

The common types of cross-validation are presented in the following:

4.2.1.1 ***Holdout validation***

The simplest method of cross-validation is holdout cross-validation in which the data set is divided in two subsets, one called training set and the other test set or validation data. Number of samples which are used for validation, is generally less than a third of initial data. In this method, first a function is fitted by function approximator using only the training data, then this new function predicts the outputs using data in the testing set which has never seen by the function. To evaluate the model we compute mean absolute test set error by accumulating the errors made by this function. Mean average error and root mean square error are two common error metric that usually used for this mean that estimated variance and standard deviation of the cross validation. The evaluation may depend on the data points which are placed in the training set and test sets. This may cause different in the evaluation depending on how the data is divided.

4.2.1.2 ***K-fold cross-validation***

On way to improve the hold out method is K-fold cross validation in which data is separated in K subsets. In the other words the holdout method is repeated k times or k folds. In the time of validation, in each step, one of the k subsets is chosen as test set and evaluate the model which achieve using the other $k-1$ subsets as training set. In fact, each of the k subsamples used once as the validation data and each data point places in a test set exactly once and $k-1$ times in a training set. To have a single estimation, we can find the average error from k error results. This approach profit the advantage of not having sensible to how the data are divided. Generally, the variance of the final estimate is decreased when the value of k is increased. The disadvantages is that we have k times more computation comparing with holdout method.

4.2.1.3 ***Leave-one-out cross-validation***

This method of cross-validation is like a k-fold cross-validation but the number of data point N is given to k. In the other words, the function approximator is trained N separate time and in each time it is trained using all data except one and validate using the left data point. For evaluating the model we can use the averaging of the errors.

4.2.2 Likelihood cross validation

On a network, there are many nodes which have estimated their models using the data on that node. This could have been done by EM-algorithm or other estimating algorithms. Some models can be estimated well but some other may not be estimated well. The estimation process may be affected by two factor:

1. data : data is the most important factor for having a good model. If we have small number of data, or we have not good examples for estimation, the risk of having a bad model will be increased.
2. estimation algorithm : the algorithm for estimating a model using data, is very important for creating good models. For example EM-algorithm is very sensitive to the initial status. It may fall in the local minimum in the estimating process thus may not find a proper model although we have good data.

To improve the efficiency of the pattern recognitions on the network, we need to find best models on the network. By the best model we mean the models which have the most likelihood to the whole data on the network. Likelihood of data x when we have a Gaussian mixture model parameters is defined as:

$$P(x|GMM) = \sum_{i=1}^{K} \pi_i \mathcal{N}(x|(\mu_i, \Sigma_i) \tag{4.1}$$

If we have a set of data $X = x_1, x_2, \ldots, x_M$, we use the mean likelihood of all data points in a data set as total likelihood of a model to that data set:

$$P(X|GMM) = \sum_{j=1}^{M} \frac{P(x_j|GMM)}{M} \tag{4.2}$$

where M is the number of data.

As the data is distributed on the network, each node have a part of it. The multimedia data are huge and we can not send them on the network to collect whole data in one node. So it is not easy to find the models which have the most likelihood to the whole data.

The size of model parameters are small, thus we can send them at low cost over the network. For comparing the models, it is good idea to find the likelihood of each model to the data of other nodes by sending them to other nodes on the network. We need a system to score the models on the nodes. The cross-validation can be used for this mean. In fact cross-validation is an approach to find better models. We can use this way to verify the model of each node with the data of the other nodes. We consider each node and its neighbors as a region. In this configuration each region has intersection with other regions. In a region with K nodes, we have K data set and K models thus we use a K-fold cross-validation to test the models. Each node compare its model to other $K - 1$ data set which exist on the region. For example figure 4.1(a) shows a region of 4 nodes. Each node in the region sends its model parameters to all nodes in that region and they compute the likelihood value of the received model to their the data (figures 4.1(b), 4.1(c), 4.1(d), 4.1(e)). If the new model produce more likelihood to the data of the node (comparing with its model)the score for the sent model will be increased. The score of each model, in fact, shows the number of nodes on the network in which this model produces better likelihood than its own models. The higher scores show the better models. There are also some nodes with the same score value. To find the best model among the models which all have the best score value, the sum of likelihoods of each model to the all data sets in the region can be used as its score and the model with higher score (likelihood) will be chosen.

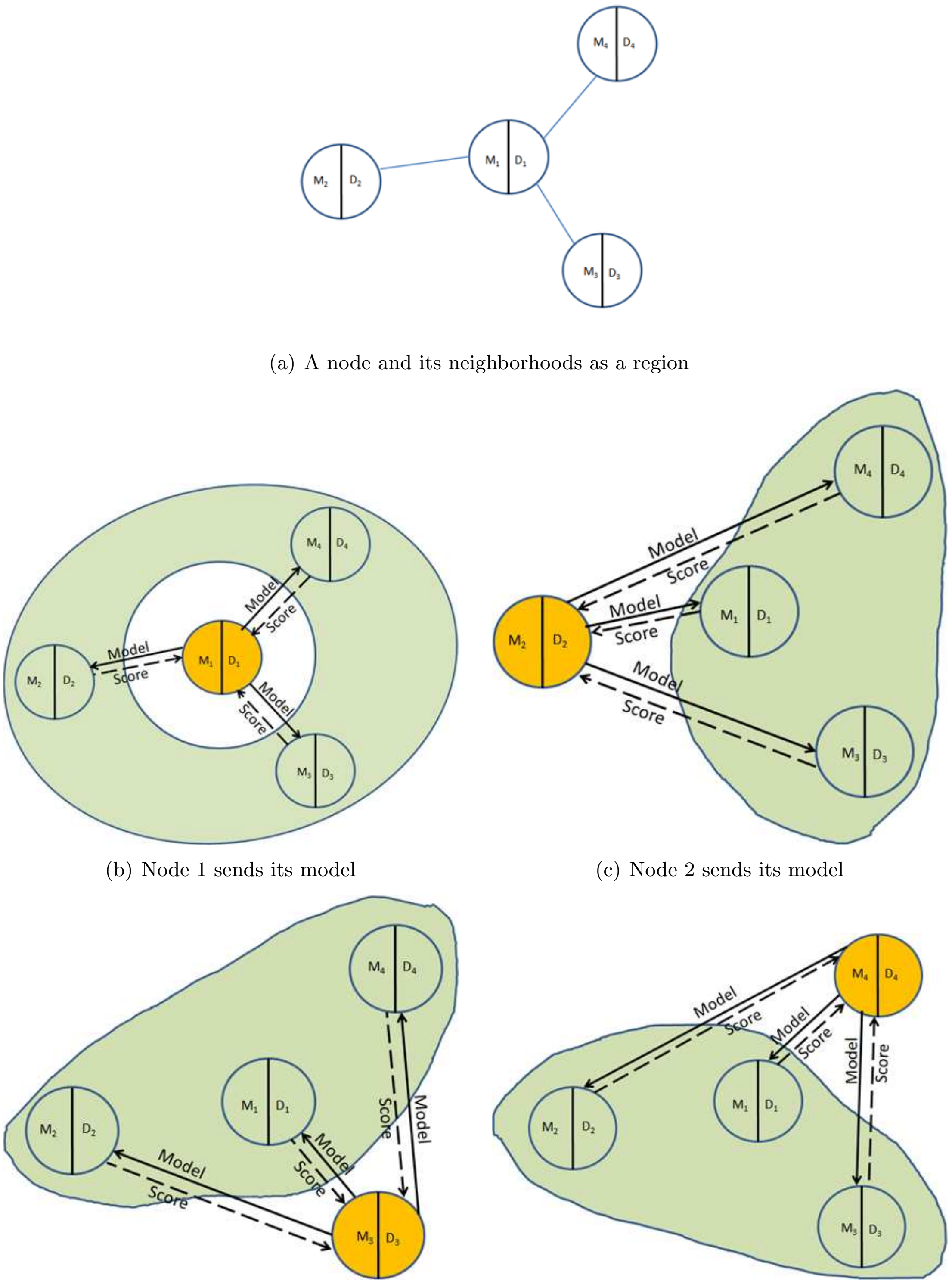
We can summarize our cross-validation algorithm as follows: first there exist two state for the nodes, active states and passive state. When a node is place on active state, it sends its model to the other nodes and when in passive state it receives the model from other node to compute a score for it. The algorithms for this two states are respectively shown in algorithms 8 and 9.

4.3 Improving models through model aggregation

The algorithms described in section 4.2.2 can find a locally best model in the region according to the nodes in the region (neighborhoods of a node). After finding the locally best model, the question is that if we can find better models by merging the models in that region. In theory we can first concatenate all the components of all nodes and then reduce it. We mention two problem for that:

1. the experiments have shown that the risk of falling in a local minimum will be increased.
2. the bad models can effect on the result.

So we decided to concatenate-reduce the models using a gossiped manner. So two nodes will be chosen randomly in the region and their components will be concatenated. As we don't like to increase the model complexity, we limit the number of components in reducing phase to the



(a) A node and its neighborhoods as a region

(b) Node 1 sends its model

(c) Node 2 sends its model

(d) Node 3 sends its model

(e) Node 4 sends its model

Figure 4.1 – Cross validation for the models in a region

Algorithm 8 Cross-validation algorithm on a region for an active node

```
1. Find the other nodes in the region which has been chosen randomly. If there is $K$ Nodes in the region, set the number 1 to $K-1$ to them.
for $i$ : from 1 to $K-1$ do
  a. Send the model parameters to node $i$
  if the score returned from the node $i$ is equal to 1 then
    add 1 to the score of this model
  end if
end for
2. Send the score value to the other nodes.
3. Find the model with maximum score (best model).
4.(For the nodes that have the same maximum score)
if There is more than one model with maximum value of score then
  Find the model which has maximum likelihood to whole data on the region (the same algorithm but using likelihood value as the score)
end if
4. (Only for the node contains best model) Send the best local model to all of the nodes in the region.
```

Algorithm 9 Cross-validation algorithm on a region for a passive node

```
1. Receive the parameters of the model
2. Find the $L_1$= likelihood of the received model as equation 4.2
3. Find the $L_2$= likelihood of the own model as equation 4.2
if $L_1 > L_2$ then
  Set S = 1 (score for new model)
else
  Set S = 0
end if
4. return S (score for received model) and $L_1$ (likelihood values of the received model) to the sender node
```

number of components of the best model found in previous phase. So to find the new models, the concatenated mixture will be reduced to all number of components smaller than the number of components of the best model found in the previous phase.

For comparing these new models and old locally best model, we need a criteria. The AIC criteria is defined as :

$$AIC(D|M) = -2logp(D|\theta) + 2\nu \tag{4.3}$$

where mixture M is defined by a parameter vector θ, $p(D|\theta)$ is the likelihood of the data for this model, ν is the number of independent parameters in the mixture. In our case, we need to find the likelihood of the new model to all of the data sets on the region and we do not wish that the likelihood be decreased. This conducted us to use the following conditions for selecting the new model as the best locally model:

Algorithm 10 The cross-validation with merging algorithm in a region

```
1. Find the model which have the best local likelihood value with the whole data of this region
using cross-validation as algorithm 8.
2. Set K = Number of nodes in the region.
for $i$ : from 1 to $K^2$ do
  a. Select at random two nodes of the region.
  b. Concatenate the models of the two selected nodes.
  for $j$ : from 1 to number of components in the previous best local model do
    Reduce the concatenated model to the $j$ components.
    if $\sum_{i=1}^{K}(log(p(New\,Model|\theta_i))) \geq \sum_{i=1}^{K}(log(p(Old\,Model|\theta_i)))$
    and $\nu_{NewModel} \leq \nu_{OldModel}$
    (the new model is better than the previous best local model) then
      I.Get the new model as the best local model.
      II.Replace the models of the two nodes with this new model.
    end if
  end for
end for
4. Send the best local model to all of the nodes in the region.
```

$$\sum_{i=1}^{K} \log p(New\,Model|\theta_i) \geq \sum_{i=1}^{K} \log p(Old\,Model|\theta_i), \tag{4.4}$$

$$\nu_{New\,Model} \leq \nu_{Old\,Model} \tag{4.5}$$

Where K is the number of nodes in the region. In other words the new model will be chosen in two cases: 1) its likelihood to all data of the region is better than the previous locally best model and it is not more complicated than the previous model 2) the new model has the same likelihood value as the locally best model but have less complexity. The merging process is repeated many times which we assure that the models in the region do not vary much by new merging. In our experiments we set the number of times to merge to the power of two of the number of the nodes in the region. Now we can add the merging to algorithm 8 and have algorithm 10.

4.4 Distribution of best model on the network

The definition of the regions in this work, cause that each node to be in a number of regions (if it has more than one neighborhood). Thus the regions have intersection with other regions. This property cause that the best models from different region can compete and distribute on the network.

Figure 4.2(a) shows an example part of a P2P network. Each node has joint to some other nodes (neighborhoods of the node). In figure 4.2(b) a region of the network has been chosen. At the end of process all the nodes have the same model which has been chosen as the best model in this region. Next time another region has been chosen in figure 4.2(c). There is one node in intersection of these two region which carry the best local model from previous region. So this model will contribute in competition and has the chance to gain and distributed in this new region.

4.5 Experimental results

Now we report experimental results obtained on real data. The example of distributed speaker recognition is taken throughout, but the technique directly applies to a wide range of multimedia classes. To evaluate the performance of the proposed technique in the distributed context described in the introduction of this work, a mechanism is required for propagating mixture representations within the network. In all the experiments, all of the nodes have learnt a probability density for speaker 'A' in the common feature (2nd feature cepstral from 13-dimension mel-cepstral feature space). Each node was provided with different training data from the same speaker and the duration of audio recordings was about 10 seconds, i.e. rather short for training. Independently of the technical contribution of this work, each node should provide its mixture estimated from its data. To this end, the Expectation-Maximization local optimization algorithm is employed, but with some enhancement [16] to limit local minima. Each mixture also autonomously and automatically determines its number of components. All covariance matrices in the mixture are full (rather than spherical or diagonal). A network of 15 nodes is used in this experiment. Each node is connected to the network through a number of nodes.To evaluate the capability of mixtures on the network to model data from the class of interest (here, a speaker), the classical marginal likelihood is selected [114]. For comparing the efficiency of the model on the whole data on the network, we have chosen two parameters: 1) likelihood to whole data 2) number of components of the models.

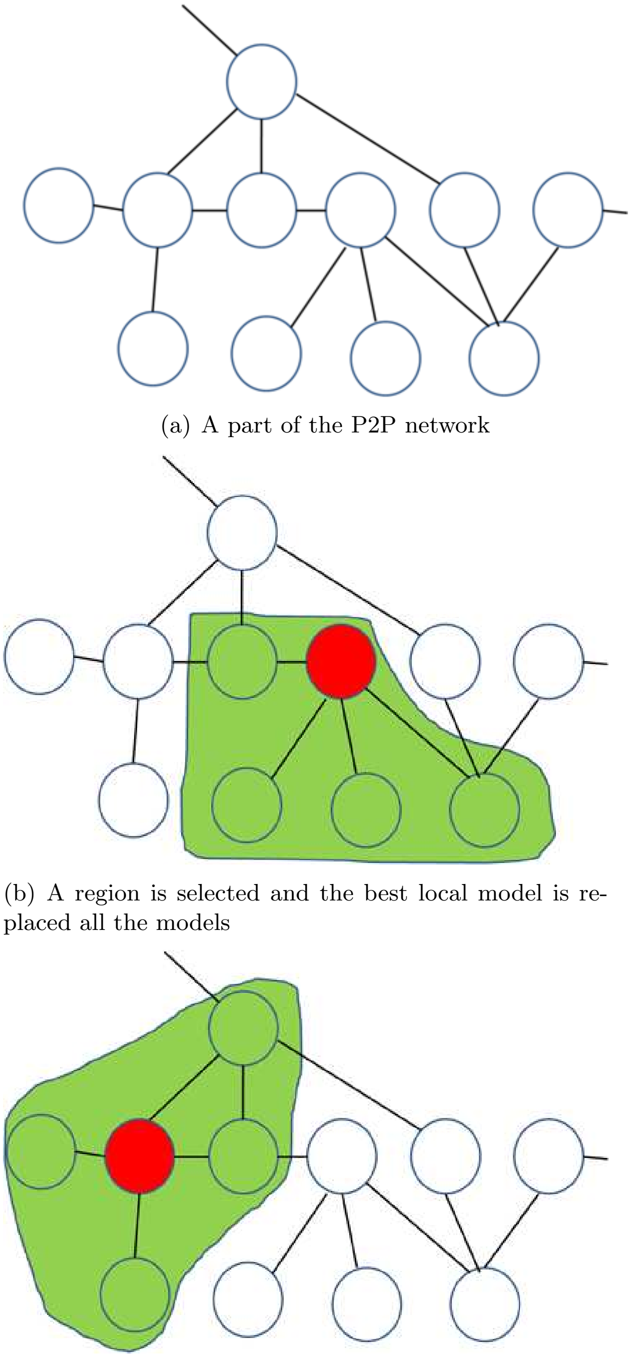
In all of the figures, the pointed line named "best model" relates to the model which have the most likelihood to the whole data before executing the algorithm. Also, the "direct model" relates to the model which has been achieved using whole data on the network. The direct model is estimated with the number of components equal to the mean number of components of all the models on the network.

We have accomplished three experiments: in the first experiment, we found each best local model in each region by only merging the models existing in that region. In the second experiment only cross-validation method was used to find the best local model through the algorithm described in section 4.2.2 and in third experiment we added the merging to the cross validation algorithm as described in section 4.3. Each experiment has been repeated 10 times.

Figure 4.3 shows the result for merging only: in the selected region, the models has been chosen in gossip manner, merged and reduced to a new model. The figures 4.3(a) and 4.3(b) shows that the likelihood value is a bit better than the direct model. Also, the mean number of components is lower than the direct model that means it has lower complexity comparing to the direct model.

Figure 4.4 shows the experiment which we have used only cross-validation algorithm. In this case, all the nodes in all experiments (10 repeat) find the best model (the model which have the best likelihood to whole data) (figures 4.4(a) and 4.4(b)). But figure 4.4(c) shows that this model have many components so its complexity is high.

Figure 4.5 is related to the experiment which we have added the merging to the previous experiment. In this experiment, after cross-validation in the chosen region, we tried to find better models in that region (models which have more likelihood with the same or fewer number of



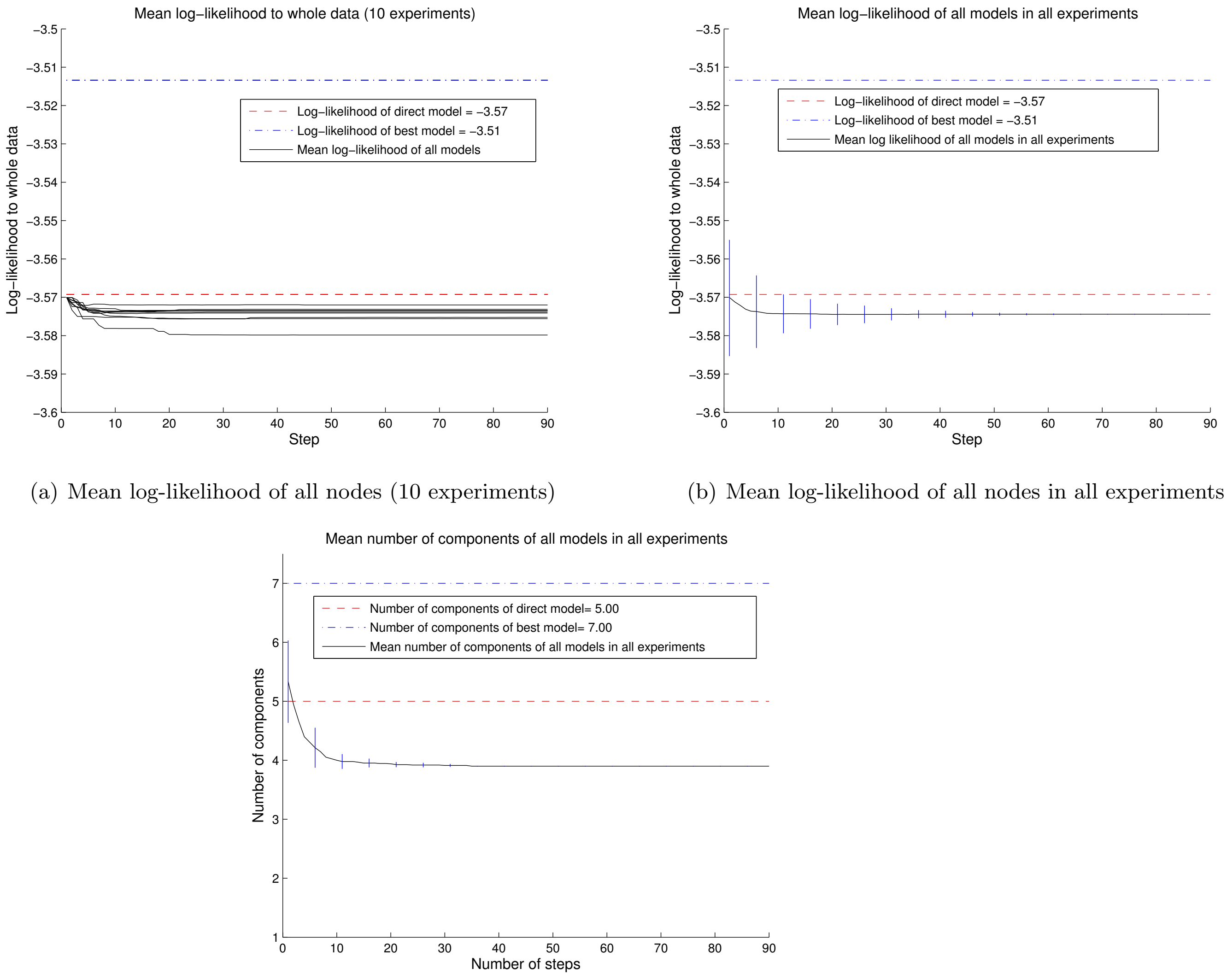
(a) A part of the P2P network

(b) A region is selected and the best local model is replaced all the models

(c) Another region is selected. The local best model from previous region has chance to compete with the models in this region

Figure 4.2 – Distributing the best model on the network

components). These new models are created using merging the models and reducing them to the models with fewer number of components comparing to the best local model. In fact we search the model which have the most likelihood value to the all data on the region but has equal or fewer number of components. We see that the algorithm can find such models and it has found the models with more likelihood than the ”best model” (figures 4.5(a) and 4.5(b)) which has fewer components (In figure 4.5(c) the mean number of components of 10 repeats is a bit more than 2). On the other words the algorithm has found the model with more likelihood and less complexity comparing to the pure cross-validation algorithm.

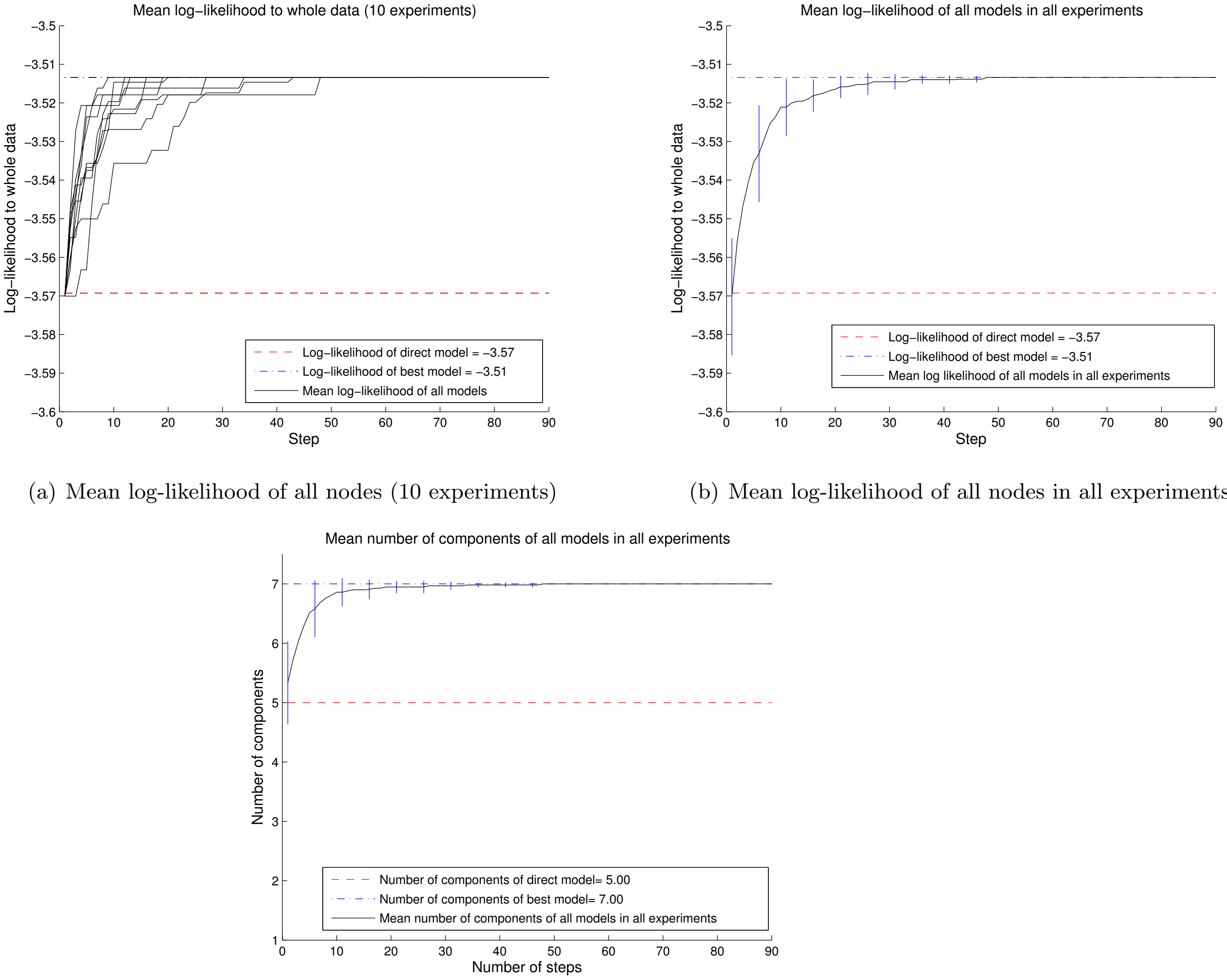


(a) Mean log-likelihood of all nodes (10 experiments)

(b) Mean log-likelihood of all nodes in all experiments

(c) Mean number of components

Figure 4.3 – Log-likelihood and components number in models in merging approach

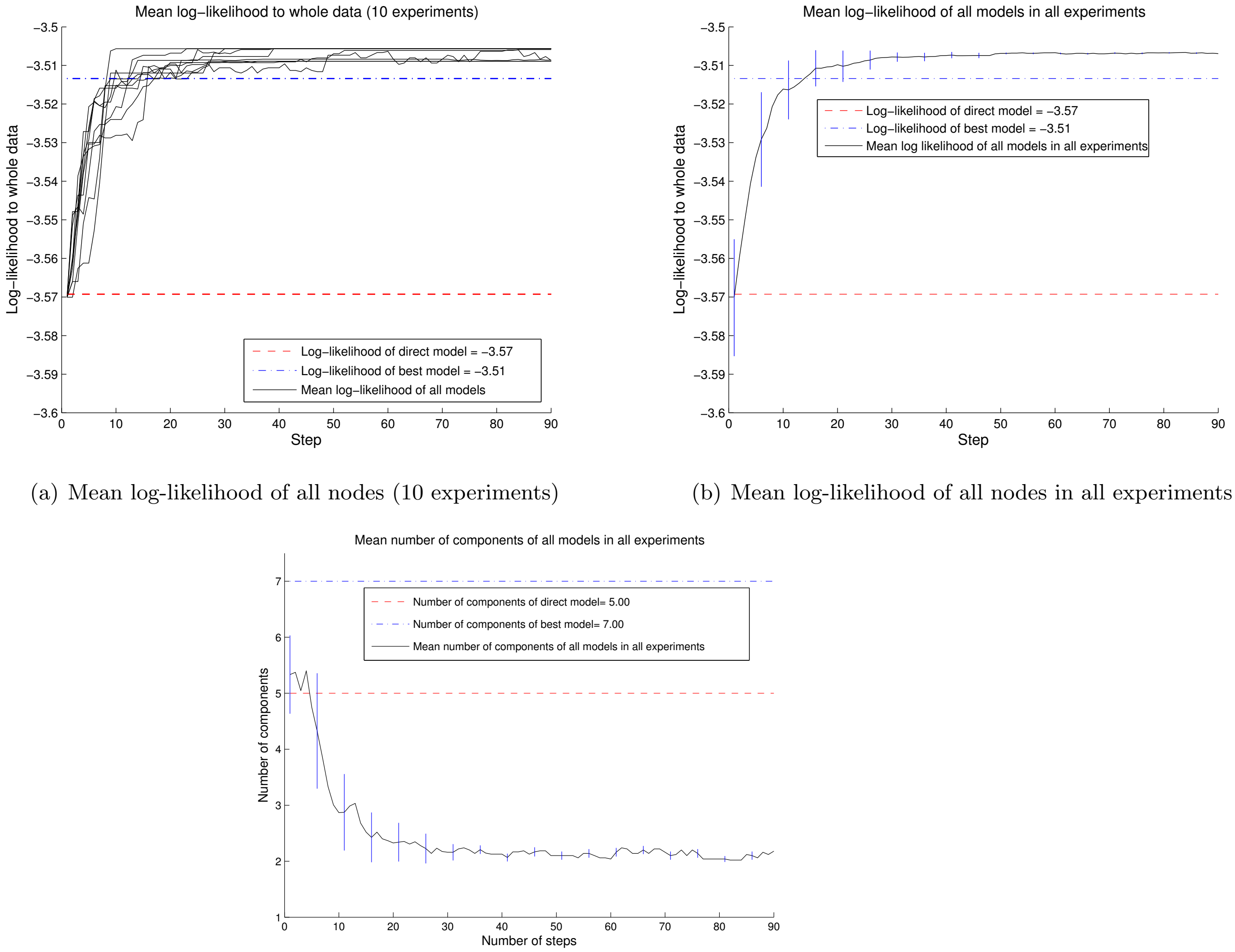


(a) Mean log-likelihood of all nodes (10 experiments)

(b) Mean log-likelihood of all nodes in all experiments

(c) Mean number of components

Figure 4.4 – Log-likelihood and components number in models in cross-validation approach



(a) Mean log-likelihood of all nodes (10 experiments)

(b) Mean log-likelihood of all nodes in all experiments

(c) Mean number of components

Figure 4.5 – Log-likelihood and components number in models in cross-validation with merging approach

4.6 Conclusion

In this chapter, we have proposed an approach to improve on the algorithm of model aggregation which we had proposed in chapter 3. In the new algorithm, a technique was proposed to evaluate the models which are distributed on the network , compare and find the best mixtures or even find the new models which are better (have more likelihood or less components) from merging the other models at low computation cost. In this algorithm we need few information to transfer on the network as each node only sends the model parameters over the network. In all experiments using the new algorithm, the mean likelihood to whole data, converged to the likelihood value more than the likelihood of the model which had the best likelihood on the network.

CHAPTER 5

Decentralized learning of a Gaussian Mixture using variational Bayes-based aggregation

In previous chapters we presented a distributed statistical estimation technique, with its motivation form, and application to multimedia content-based indexing. The contribution was a scheme for estimating a multivariate probability density, in the case where this density takes the form of a Gaussian mixture model. They have of broad applicability for multimedia feature modeling. Assuming independently estimated mixtures, we propagate their parameters in a decentralized fashion (gossip) in a network, and aggregate GMMs from connected nodes, to improve estimation.

In this chapter, as an improvement through a change of principle over previous work, aggregation is achieved through Bayesian modeling of the GMM component grouping problem and solved using a variational Bayes technique, applied at component level. This determines, through a single, low-cost yet accurate process, assignments of components that should be aggregated and the number of components in the mixture after aggregation. Because only model parameters are exchanged on the network, computational and network load remain very moderate. The scheme is demonstrated on the task of speaker recognition.

5.1 Introduction

As we said in chapter 2, a standard method to learn a model is maximum likelihood(ML). This method estimates optimal values for the model parameters within a fixed graph structure from a given data set. There are three main problems with ML learning:

1. it may have suboptimal generalization performance when it produce a model that overfits the data.
2. more complicated graphs assigns a higher likelihood to the data so it can not be used for learning the structure of the graphs.
3. it is computationally tractable only for a small class of models.

The Bayesian framework can solve, generally, the first two problems. In this framework we consider an ensemble of models that are characterized using a probability distribution over all possible parameter values and structures. Given a dataset, the distribution over the ensemble of models will be computed instead of learning a single model. Thus to enhance the generalization performance, this framework uses model uncertainly. In addition, a lower posterior probability is assigned to the complex models to penalize them and find the optimal structure.

Unfortunately, we need to integrate over models in this framework whose computations can seldom be performed exactly. Therefore, approximation must be made for this mean. The major schemes are Markov chain Monte Carlo(MCMC) methods and Laplace approximation. MCMC attempts to achieve exact results but typically requires vast computational resources. It has lower complexity of $O(m^2N)$ where m is the number of parameters and N the dataset size, but it is a good approximation only when $N/m \rightarrow \infty$.

Variational methods can be used as approximation methods in very many settings. In fact, the term of "variational methods" refer to a large collection of optimization methods. But in all of them, a complex problem is converted to a simpler problem. The simpler problem is usually achieved by decoupling of the degrees of freedom in the initial problem. This can be performed by expansion of the problem to have additional parameters which called variational parameters, and must be fit to the problem at hand. The solution to variational problems are usually in terms of some fixed point equations which have necessary conditions for optimality. These are analogous to setting the gradient to zero in ordinary function optimization. In last decade, many problems of inference and estimation in large density connected graphical probabilities models that can not solved by exact probabilistic calculations, are solved by variational approaches (for example [89]). There are two properties that cause the success of variational solutions: 1) probabilistic inference problems can be formulated as a variational problem, 2) the achieved variational optimization admit approximate solution. In fact variational formulation facilitate finding the approximate solution. For example to solve the extremum problems that involve an unknown function, we can restrict their space of admissible functions (for example in terms of a finite number of basis function) to approximate them or in the context of probabilistic calculations, analogous restrictions are usually achieved by factorization.

Attias [8] presented the variational Bayes framework also called ensemble learning, for computations in graphical models. The author has applied variational inference to the general problem of Bayesian parameter estimation. In this types of solutions, we usually treat parameters as new nodes in a graphical model [81] and then apply Bayesian inference on it. These probabilistic inference problems are often intractable, thus variational approximations can be helpful for solving them. In fact, variational Bayesian methods, are techniques to facilitate analytical calculations of posterior distributions over the hidden variables, parameters and structures by approximating the integrals which raised in Bayesian statistics.

The rest of this chapter is organized as follows. Section 5.2 present the variational Bayesian learning and then variational mixture of Gaussian. In section 5.3 we study about the problems of estimating the reduced mixture using virtual sampling and we will find an algorithm for grouping the components to reduce them. In section 5.4 we propose a variational Bayesian solution for GMM reduction using a virtual sampling and then we use this approach to aggregate the mixture of Gaussians to find the better models. The experimental results are in the section 5.5 and section 5.6 is a conclusion about this chapter.

5.2 Variational Bayesian learning

The maximum likelihood and maximum penalized likelihood setting which discussed in chapter 2, cannot automatically inferred the model structure. In these learning algorithms, the likelihood function is unbounded, thus, they favor models without looking to their complexity and therefor they can not perform automatically model selection. The Bayesian setting profit the advantage of having the model complexity in the problem statement. In this section, we present variational Bayesian leartning.

5.2.1 Bayesian learning or Bayesian inference

In Bayesian inference, the observations or evidence are used to calculate new probability that a hypothesis may be true that is useful for decision making under uncertainty. The uncertainly is the main advantage of Bayesian inference. This approach permit us to find the optimal model complexity without using the statistical resampling techniques.

As we discussed previously, the posterior distribution of the parameters is used also in MAP learning, but in Map learning, predictions are performed based on point-estimates and does not properly deal with the uncertainty on the parameters. However in Bayesian learning, the parameters are seemed as (latent) random variables and we have better uncertainly on the parameters when we use their posterior distribution for constructing the predictive distribution:

$$p(x) \approx p(x|X) = \int p(x|X,\Theta)p(\Theta|X)d\Theta = \int p(x|\Theta)p(\Theta|X)d\Theta \tag{5.1}$$

where it is assumed that the prediction x is independent of X given Θ.

There are two stage in Bayesian learning : model fitting and model selection. In the model fitting stage, we have given the data observation X and by supposing that the model structure M is fixed, the parameters Θ are learnt. To update the prior belief on the parameters and compute the posterior distribution over the parameters, the Bayes rule can be applied:

$$\underbrace{p(\Theta|X,M)}_{\textit{Posterior}} = \frac{\overbrace{p(X|\Theta,M)}^{\textit{Likelihood}}\overbrace{p(\Theta|M)}^{\textit{Prior}}}{\underbrace{p(X|M)}_{\textit{Evidence}}} \tag{5.2}$$

Which allows us to quantify our uncertainly about parameter values after observing the data. If we have latent variables, the likelihood is the incomplete data likelihood which defined as

$$L(\Theta|X) \equiv p(X|\Theta) = \prod_{n=1}^{N} \int p(x_n, z_n|\Theta)\, dz_n. \tag{5.3}$$

In the equation 5.2, the evidence is in fact the probability of observing the data given a particular model M. In the model fitting stage or first level of inference, evidence is not important but it has an important role in the model selection stage (second stage of inference).

There are some assumption or prior knowledge about the model structure in the beginning of a Bayesian approach for learning that is represented in the form of a prior probability distribution over model structures. This prior distribution is updated by the data and a posterior distribution over models and parameters is achieved. More formally, if we suppose the prior distribution over the model structures is $p(M)$ and for each model structure the prior distribution over parameters is $P(\Theta|M)$, after observing the data set X we can use Bayes'rules to compute the posterior distribution over models:

$$p(M|X) \propto p(X|M)p(M) \tag{5.4}$$

The most probable model or model structure is the one that maximizes $P(M|X)$. The prior $p(M)$ is often chosen to be uniform because all model has the same priority and we don't favor one model to another model. Thus we can rank the models by their evidence ($p(X|M)$). Now, we compute the model evidence by integrating out the model parameters:

$$p(X|M) = \int p(X|\Theta, M)p(\Theta|M)d\Theta \tag{5.5}$$

This integral is usually intractable but we can use variational inference which discussed in the next section.

5.2.2 Variational inference or variational Bayes

To approximate the integrals in for Bayesian learning, variational methods can be used [19, 8]. The basic idea is to suppose that the hidden states and parameters are independent then approximate the distribution over both parameters and hidden states using a simpler distribution. In fact in this approaches, the posterior distribution over a set of latent variables $Z = \{z_1 \dots z_n\}$ and parameter Θ given some data X is approximated by a variational distribution:

$$P(Z, \Theta|X) \approx Q(Z, \Theta). \tag{5.6}$$

$Q(Z, \Theta)$ is variational a distribution which has a simpler form than $P(Z, \Theta|X)$. To perform this, in fact the range of functions over witch the optimization is performed will be restricted.

Now applying of variational optimization into inference problem is presented in more detail. Assuming a fully baysian model that all parameters Θ have a prior distribution. The observed data are a set of N independent, identically distributed data, that denote with $X = \{x_1, x_2, \dots, x_n\}$. There are also latent variables $Z = \{z_1, z_2, \dots, z_n\}$. The parameters can be treat as latent variables and the probabilistic model is a joint distribution $p(X, Z, \Theta)$. The goal is to find an approximation for posterior distribution $p(Z, \Theta|X)$ and model evidence $p(x)$.

Using Jensen's inequality we can result that for any auxiliary distribution $q(Z, \Theta)$, the logarithm of the evidence $p(x|M)$ can be lower bounded :

$$\begin{aligned}
\ln p(X|M) &= \ln \iint p(X, Z, \Theta|M)\, dZ\, d\Theta && (5.7)\\
&= \ln \iint q(Z, \Theta) \frac{p(X, Z, \Theta|M)}{q(Z, \Theta)}\, dZ\, d\Theta && (5.8)\\
&\geq \iint q(Z, \Theta) \ln \frac{p(X, Z, \Theta|M)}{q(Z, \Theta)}\, dZ\, d\Theta. && (5.9)
\end{aligned}$$

We can decompose the log marginal probability using

$$\ln p(X|M) = F_M(q(Z,\Theta)) + KL[q(Z,\Theta)\|p(Z,\Theta|X,M)] \tag{5.10}$$

where we have defined

$$F_M(q(Z,\Theta)) = \iint q(Z,\Theta)\ln\{\frac{p(X,Z,\Theta|M}{q(Z,\Theta)}\}\, dZ\, d\Theta \tag{5.11}$$

$$KL[q(Z,\Theta)\|p(Z,\Theta|X,M)] = -\iint q(Z,\Theta)\ln\{\frac{p(Z,\Theta|X,M}{q(Z,\Theta)}\}\, dZ\, d\Theta \tag{5.12}$$

The lower bound $F_M(q(Z,\Theta))$ can be maximized by optimizing it with respect to the distribution $q(Z,\Theta)$. This is equivalent to minimizing the KL divergence. If we allow any possible choice for $q(Z,\Theta)$ then the maximum of the lower bound occurs when KL divergence vanishes. This occurs when $q(Z,\Theta)$ is equal to the posterior distribution $p(X|Z,\Theta,M)$. However, the model is usually such that working with the true posterior distribution is intractable.

In variational Bayes (VB), an approximate posterior is chosen in such a way that the lower bound becomes tractable by considering a restricted family of distribution $q(Z,\Theta)$ and then seek the member of this family for which the KL divergence is minimized. The goal is to find a restricted family that comprise only tractable distributions, while at the same time the family should be sufficiently rich and flexible that we have good approximation to the true posterior distribution.

5.2.2.1 ***factorized distributions***

Factorizing distribution is a way to restrict the family of distribution $q(Z,\Theta)$. Assuming that the elements of Z are partitioned into disjoint groups that are denoted by Z_i where $i = 1,\dots,N$. We then suppose that the variational posterior distribution q factorizes with respect to these groups, so we have:

$$q(Z,\Theta) = q_Z(Z)q_\Theta(\Theta) = (\prod_{i=1}^{N} q_{z_i}(Z_i))q_\Theta(\Theta) \tag{5.13}$$

That is a consequence of the data X being i.i.d (independent and identically distributed). Thus, given the observed data, there is independency between the parameters and the latent variables for variational approximation of the joint posterior. In fact, by decoupling the degrees of freedom in the initial problem, it is converted into a simpler problem. Now we can describe the lower bound on the log-evidence such as follows

$$\ln p(X|M) \geq \iint q_Z(Z)q_\Theta(\Theta)\ln\frac{p(X,Z,\Theta|M)}{q_Z(Z)q_\Theta(\Theta)}\, dZ\, d\Theta \tag{5.14}$$

$$\equiv F_M(q_{z_1}(z_1),\dots,q_{z_N}(z_N),q_\Theta(\Theta)) \tag{5.15}$$

The variational Bayesian approach iteratively maximize the bound F with respect to the free distributions, $q_z(Z)$ and $q_\Theta(\Theta)$ which leads to the equations like EM update and called variational Bayesian EM (VBEM) [13]:

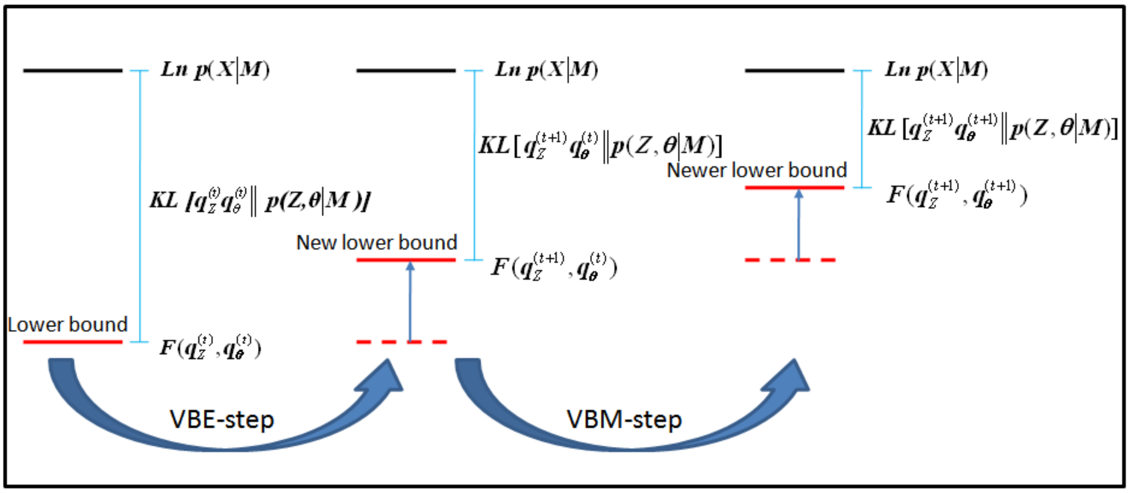

$Ln\ p(X|M)$
$KL\ [q_Z^{(t)} q_\theta^{(t)} \| p(Z,\theta|M)]$
Lower bound
$F(q_Z^{(t)}, q_\theta^{(t)})$
VBE-step
$Ln\ p(X|M)$
$KL[q_Z^{(t+1)} q_\theta^{(t)} \| p(Z,\theta|M)]$
New lower bound
$F(q_Z^{(t+1)}, q_\theta^{(t)})$
VBM-step
$Ln\ p(X|M)$
$KL\ [q_Z^{(t+1)} q_\theta^{(t+1)} \| p(Z,\theta|M)]$
Newer lower bound
$F(q_Z^{(t+1)}, q_\theta^{(t+1)})$

Figure 5.1 – The variational Bayesian EM (VBEM) algorithm. In both the VBE-step and VBM-step, the lower bound on the log-evidence is increased. Subscript t denotes the iteration number.

$$VBE - step : q_{z_n}(z_n) \propto exp(\mathbb{E}_\Theta\{\ln p(x_n, z_n|\Theta, M)\}), \forall n. \tag{5.16}$$

$$VBM - step : q_\Theta(\Theta) \propto p(\Theta|M)exp(\mathbb{E}_Z\{\ln L_c(\Theta|X, Z, M)\}). \tag{5.17}$$

where $\mathbb{E}_\Theta\{.\}$ and $\mathbb{E}_Z\{.\}$ are respectively the expectation with respect to $q_\Theta(\Theta)$ and $q_Z(Z)$, and $L_c(\Theta|X, Z, M)$ is the complete data likelihood defined as:

$$L_c(\Theta|X, Z, M) \equiv p(X, Z|\Theta, M) = \prod_{n=1}^{N} p(x_n, z_n|\Theta, M). \tag{5.18}$$

VBEM is guaranteed to monotonically increase. The VBE-step maximizes the lower bound with respect to $q_{z_n}(z_n), \forall n$ and the VBM-step maximizes the lower bound with respect to $q_\Theta(\Theta)$.

In VBEM, there are no distinction between hidden variables and parameters in VBEM step except that the number of variables is increased with the number off data, but number of parameters is not changed. As illustrated in figure 5.1, both VBE-step and VBM-step minimize the kl-value between the variational posterior $(q_Z(Z)q_\Theta(\Theta))$ and the true joint posterior of the hidden variables and parameters give data $(p(Z, \Theta|X, M))$ which consequences maximization of lower bound on the log-evidence $(\ln p(X|M))$:

$$F_M(q_{z_1}(z_1), \ldots, q_{z_N}(z_n), q_\Theta(\Theta)) \tag{5.19}$$

$$= \ln p(X|M) - KL[q_Z(Z)q_\Theta(\Theta)\|p(Z, \Theta|X, M)]. \tag{5.20}$$

The factorized posterior is an approximation of true posterior. It tries to make the bounds as tight as possible to have a good approximation in terms of KL divergence. The VBEM algorithm should start from an initial prior distribution over the parameters. General the conjugate distribution to the exponential family is chosen as prior so that the posterior distribution has the same functional form as the priors. The function $f(x|\Theta)$ is conjugate to the the prior $p(\Theta)$, if the posterior $q(\Theta|x)$ has the same functional form as $p(\Theta)$ or $p(\Theta) \propto f(x|\Theta)p(\Theta)$. This facilitate the computations in VBEM and learning in this framework consist only updating the parameters.

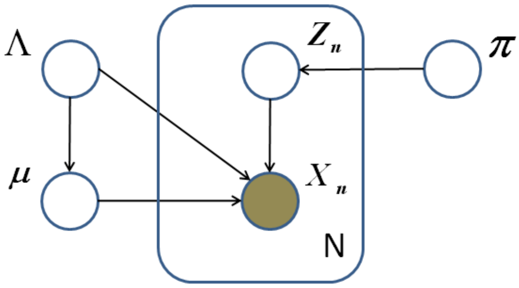


Figure 5.2 – Graphical representation of the Bayesian mixture of Gaussian model, in which the box denotes a set of N i.i.d observation. Here μ denotes $\{\mu_k\}$ and Λ denotes $\{\Lambda_k\}$.

5.2.3 Variational mixture of Gaussian

Uncertainly on the parameters of the model is taken into account in the Bayesian inference that is its the main advantage. The optimal model complexity in this approach can be achieved without using the statistical resampling techniques. In previous section, we have investigated the variational Bayesian approach and we saw that posterior distribution for the parameters and latent variables can be learned using an EM-like algorithm. In this section, variational Bayes (VB) is applied to the GMM. As we described in chapter 2, GMM can be supposed as a latent variable model that we don't know each data point is generated by which component. In this view, for each observation x_n we have a corresponding latent variable z_n comprising a 1-of-K binary vector with elements z_{nk} for $k = 1, \ldots, K$. As before, we denote the observed data set by $X = \{x_1, \ldots, x_N\}$ and similarly denote the latent variables by $Z = \{z_1, \ldots, z_N\}$. We can write the conditional distribution of Z, giving the mixing coefficient π, in the form

$$p(Z|\pi) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}} \tag{5.21}$$

The corresponding graphical model is shown in figure 5.2.

Since the model parameters $\Theta_N = \{\pi, \mu, \Lambda\}$ are treated as random variables,they appear as nodes in the graph where $\pi = \{\pi_k\}$, $\mu = \{\mu_k\}$ and $\Lambda = \{\Lambda_k\}$. We work in terms of precision matrices rather than covariance matrices as this somewhat simplifies the mathematics. The conditional distribution of the observed data vectors, given the latent variables and component parameters can be written as:

$$p(X|Z, \mu, \Lambda) = \prod_{n=1}^{N} \prod_{k=1}^{K} \mathcal{N}(x_n|\mu_k, \Lambda_k^{-1})^{z_{nk}} \tag{5.22}$$

Now we want to find the Bayesian framework of GMM. The model evidence (incomplete data likelihood) in GMM, can be computed by integrating over the latent variables Z and the parameters Θ_N:

$$p(X|M) = \sum_{Z} \int p(X, Z, \Theta_N|M)\, d\Theta_N \tag{5.23}$$

But it can not be computed directly and is intractable. To compute this quantity, we first find a lower bound for the value of log-evidence for any arbitrary density $q(Z, \Theta_N)$. It can be found by Jensen's inequality as follows:

$$\ln p(X|M) \geq \ln p(X|M) - KL[q(Z, \Theta_N) \| p(Z, \Theta_N|X, M)] \tag{5.24}$$

The maximum value of right of the above equation is when $q(Z, \Theta_N)$ is equal to the joint posterior of the latent variables and the parameters or $p(Z, \Theta_N|X, M)$. Now we assume that latent variable and parameters are independent and factorize the joint posterior $q(Z, \Theta_N)$ (to able to find a VB learning):

$$q(Z, \Theta_N) = q_Z(Z) q_{\Theta_N}(\Theta_N). \tag{5.25}$$

With this assumption the lower bound on the log-evidence is tractable. It able us to find an algorithm for learning the parameters of VB-GMM. The optimized values for parameters and latent variables are when the KL divergence between the true posterior $p(Z, \Theta_N|X, M)$ and the variational posterior $q_Z(Z)q_{\Theta_N}(\Theta_N)$ become minimum. As data X are i.i.d the posterior $q_Z(Z)$ can be also factorized. This lead us to un iteratively EM-like rules such that are said in section 5.2.2:

$$VBE - step : q_{z_n}(z_n) \propto \exp(\mathbb{E}_{\Theta_N}\{\ln p(x_n, z_n|\Theta_N, M)\}), \forall n. \tag{5.26}$$

$$VBM - step : q_{\Theta_N}(\Theta_N) \propto p(\Theta|M) \exp(\mathbb{E}_Z\{\ln L_c(\Theta_N|X, Z, M)\}). \tag{5.27}$$

Where the expectation with respect to $q_Z(Z)$ and $q_{\Theta_N}(\Theta_N)$ are respectively denoted by $\mathbb{E}_Z\{.\}$ and $\mathbb{E}_{\Theta_N}\{.\}$. Now we compute the VBE-step and VBM-step for GMM. When we have a Gaussian mixture model, the complete data likelihood is as following:

$$L_c(\Theta_N|X, Z, M) = \prod_{n=1}^{N} p(x_n, z_n|\Theta_N, M) \tag{5.28}$$

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}} \mathcal{N}(x_n|\mu_k, \Lambda_k)^{z_{nk}} \tag{5.29}$$

where we used the latent variable formulation of GMM and $p(x_n, z_n|\Theta_N, M)$ factorizes. Similarly we can factorize $\{q_{z_n}(z_n)\}_{n=1}^N$:

$$q_{z_n}(z_n) = \prod_{k=1}^{K} q_{z_{nk}}(z_{nk})^{z_{nk}}, \forall n. \tag{5.30}$$

This leads the VBE-step for the GMM simplifies to :

$$\rho_{nk} = q_{z_{nk}}(z_{nk} = 1) \propto exp(\mathbb{E}_{\Theta_N}\{\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Lambda_k)\}). \tag{5.31}$$

The quantities ρ_{nk} correspond to the responsibilities in ML and MAP learning. In other words, each ρ_{nk} is proportional to the posterior probability of having a component k when x_n is observed. Now, there is required to find $q_{\Theta_N}(\Theta_N)$ to able computing the VBE-step.

If we look at VBM-step, it is attractive to use the conjugate to the exponential family for the prior $p(\Theta_N|M)$ distribution which cause the posterior to be the same functional form as the prior. Thus we only need to update the parameters of the prior to have parameters of the posterior in VBM-step. As we said in chapter 2, the joint conjugate prior for the GMM is the product of a joint Dirichlet prior on the mixture proportions and Gaussian-Wishart distributions on the means and precisions of each Gaussian component:

$$p(\pi|M) = Dir(\pi|\alpha_0) = C(\alpha_0)\prod_{k=1}^{K}\pi_k^{\alpha_0-1} \tag{5.32}$$

$$p(\mu,\Lambda|M) = p(\mu|\Lambda)p(\Lambda) = \prod_{k=1}^{K}\mathcal{N}(\mu_k|m_0,(\beta_0\Lambda_k)^{-1})W(\Lambda_k|W_0,\nu_0) \tag{5.33}$$

where by symmetry we choose the same parameter α_0 for each component and $C(\alpha_0)$ is the normalization constant for Dirichlet distribution. We also choose $m_0 = 0$ by symmetry. The joint conjugate prior for the GMM is

$$p(\Theta_N|M) = Dir(\pi|\alpha_0)\prod_{k=1}^{K}\mathcal{NW}(\mu_k,\Lambda_k|\Theta_{NW_0}) \tag{5.34}$$

where $\Theta_{NW_0} = (\beta_0, m_0, \nu_0, W_0^{-1})$ are particular values for the hyperparameters. In practice, they are chosen such that broad priors are obtained. Remark the property of the conjugate prior, the joint posterior has the same functional form and contains the product of a Dirichlet and Gaussian-Wishart distributions:

$$q_{\Theta_N}(\Theta_N) = Dir(\pi|\alpha)\prod_{k=1}^{K}\mathcal{NW}(\mu_k,\Lambda_k|\Theta_{NW_k}) \tag{5.35}$$

where $\Theta_{NW_k} = (\beta_k, m_k, \nu_k, W_k^{-1})$. As the form of the posterior is known we can now compute the expectation in VBE-step. The expected value of $\mathbb{E}\{(x-m)^T A(x-m)\}$ can be computed using :

$$\mathbb{E}\{(x-m)^T A(x-m)\} = (\mu-m)^T A(\mu-m) + tr\{A\Lambda^{-1}\} \tag{5.36}$$

and supposing $x \sim \mathcal{N}(.|\mu,\Lambda)$ and given that $\mathbb{E}_{\Theta_N}\{\Lambda_k\} = \nu_k W_k$ under the Wishart prior, we can obtain the following equation:

$$\mathbb{E}_{\Theta_N}\left\{-\frac{1}{2}(x_n-\mu_k)^T\Lambda_k(x_n-\mu_k)\right\} = -\frac{\nu_k}{2}(x_n-m_k)W_k(x_n-m_k) - \frac{D}{2\beta} \tag{5.37}$$

where D is the dimensionality of the data variable x. We can obtain the VBE-step for the GMM by substituting the equation 5.37 in 5.31:

$$\rho_{nk} = q_{z_{nk}}(z_{nk}=1) \propto \tilde{\pi}_k(2\pi)^{\frac{-D}{2}}\tilde{\Lambda}_k^{\frac{1}{2}}exp(\frac{\nu_k}{2}(x_n-m_k)^T W_k(x_n-m_k) - \frac{D}{2\beta_k}) \tag{5.38}$$

where the special quantities $\tilde{\pi}_k$ and $\tilde{\Lambda}_k$ are defined:

$$\ln \tilde{\pi}_k \equiv \mathbb{E}_{\Theta_N}\{\ln \pi_k\} = \psi(\alpha_k) - \psi(\hat{\alpha}) \tag{5.39}$$

$$\ln \tilde{\Lambda}_k = \mathbb{E}_{\Theta_N}\{\ln |\Lambda_k|\} = \sum_{i=1}^{D} \psi\left(\frac{\nu_k + 1 - i}{2} + D \ln 2 + \ln |W_k|\right) \tag{5.40}$$

In these equations, $\hat{\alpha} = \sum_k \alpha_k$ and $\psi(.)$ denotes the diggama function. The responsibilities are achieved from the fact that $q_{z_n}(z_n)$ must be normalized for each data point x_n:

$$r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}} \tag{5.41}$$

That its form is similar to the responsibilities that are found for E-step of maximum likelihood learning algorithm (equation 2.31 or 2.40). Now to compute the VBM-step, we use the following equation:

$$\mathbb{E}_Z\{\ln L_c(\Theta_n|X, Z, M)\} = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\{\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Lambda_k)\}, \tag{5.42}$$

and the VBM update rules for the hyperparameters can be found after some algebra from 5.27:

$$\alpha_k = N_k + \alpha_0 \tag{5.43}$$

$$\beta_k = N_k + \beta_0 \tag{5.44}$$

$$m_k = \frac{N_k \bar{x}_k + \beta_0 m_0}{\beta_k} \tag{5.45}$$

$$\nu_k = N_k + \nu_0 \tag{5.46}$$

$$W_k^{-1} = W_0^{-1} + N_k S_k + \frac{N_k \beta_0}{\beta_k}(\bar{x} - m_0)(\bar{x} - m_0)^T \tag{5.47}$$

where we have

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{5.48}$$

$$\bar{x}_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{N_k} \tag{5.49}$$

$$S_k = \frac{\sum_{n=1}^{N} r_{nk}(x_n - \bar{x}_k)(x_n - \bar{x}_k)^T}{N_k} \tag{5.50}$$

$$\bar{\pi}_k = \frac{N_k}{N}. \tag{5.51}$$

As we said already, we used $\Lambda_k = \Sigma_k^{-1}$ for simplifying equations. Also note that the equations 5.48 to 5.50 are in fact the means of the posterior distributions which are identical to the estimation of parameters in M-step of EM algorithm. The parameter estimation of a variational posterior for a GMM is an EM-like optimization that cycle between the two steps E and M. In the variational equivalent of E-step, we use the current distributions over the model parameters to evaluate the moments in 5.37, 5.39 and 5.40 and hence evaluate $\mathbb{E}[Z_{nk}] = r_{nk}$. Then in the subsequent variational equivalent of the M step, we keep these responsibilities fixed and use them to recompute the variational distribution over the parameters using 5.35 and 5.43 to 5.47. If $N \to \infty$ the posterior collapse onto their means and $\tilde{\pi}_k = \bar{\pi}_k, \tilde{\Lambda}_k = |\Lambda_k|$ and the standard EM is recovered in the limit.

A nice property of variational mixture of Gaussian that is investigated in [7] is that if the number of data points assigned to components k is 1 or less i.e. $\bar{\pi}_k \leq 1/N$, VBEM assigns zero to $\bar{\pi}_k$. This solves a well-known problem with ordinary EM (infinite probability mass on a single data point). However, different initializations may be useful to find a good maximum since it can be multiple maxima in the variational bound.

5.2.3.1 ***Number of components in Variational Bayes Mixture Model***

The usual approaches to determination number of components (K) need to train and compare a range of models having different K values. In variational Bayesian mixture, mixing coefficient π is teated as parameters and by maximizing the lower bound with respect to π it make a point estimates of their values. This leads to re-estimation equation

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} r_{nk} \tag{5.52}$$

and the maximization is interleaved with the variational updates for the q distribution over the remaining parameters. In VB approaches, during the optimization, the mixing coefficient of components that have insufficient contribution to explaining the data are driven to zero so they are removed automatically from the model. This effect can be automatically trade-off between fitting the data and the complexity of the model. In other word, it remains only a number of components that after convergence, their expected values of the mixing coefficients are distinguishable from their priors and the complexity penalty arises from components whose parameters are pushed away from prior values. Components that take essentially no responsibility for explaining the data points have $r_{nk} \approx 0$ and hence $N_k \approx 0$. From equation 5.43 we obtain $\alpha_k = \alpha_0$ and then from equations 5.44 to 5.47 the other parameters revert to their prior values. In fact these components are fitted slightly to the data points. The expected value of the mixing coefficients in the posterior distribution in the variational Gaussian mixture model are driven from:

$$\mathbb{E}[\pi_k] = \frac{\alpha_k + N_k}{K\alpha_0 + N} \tag{5.53}$$

in which if $\alpha_0 \to 0$, for the components that have $N_k \approx 0$ and $\alpha_k \approx \alpha_0$, we have $\mathbb{E}[\pi_k] \to 0$ that shows the component plays no role in the model.

5.3 GMM component reduction using virtual sampling instead of GMM estimation from data

In this section, we try to estimate the reduced mixture using the parameters of the concatenated mixture using the idea which is discussed in [186] for hierarchical mixtures. The approach suppose a virtual sample generated from the mixture M_c instead of generating a real sample from same model. Then, we use EM similar algorithm to find the expression for the parameters of the mixture model M_r that best explain this virtual sample, and establish a closed-form relationship between these parameters and the parameters of model M_c. Nonetheless, our problem may be recast into the classical GMM-from-data setting as follows : assume data X were generated from M_c. The goal is to find M_r that maximizes the evidence $\ln p(X)$, with the constraint that the components of M_r, are linear combinations of components of M_c and so no actual sampling from M_c needs to be done.

Each model M_c and M_r consist of the mixture densities where the data is described by

$$P(X|M_c) = \sum_{k=1}^{m^c} \pi_k^c p(X|h_k^c = 1, M_c) \tag{5.54}$$

$$P(X|M_r) = \sum_{k=1}^{m^r} \pi_k^r p(X|h_k^r = 1, M_r) \tag{5.55}$$

where m^c and m^r are the number of mixture components in the M_c and M_r. π_k^c and π_k^r are the prior probability of the k^{th} component for M_c and M_r . h_k^c and h_k^r are binary variables that take value 1 if and only if the sample X was drawn from that component.

Now the problem is to compute the mixture parameters of the M_r when we have the parameters of M_c. One solution is to draw a sample from the mixture density M_c and use EM algorithm with the wished number of classes for M_r to estimate the corresponding parameters. But it would be computationally expensive. Next, we show that we can use a virtual sample to find the formula for finding M_r. For this mean, we assume N virtual sample point from M_c and called it $X = X_1, \ldots, X_{m^c}$ where each of X_i is a virtual sample set with size $N_i = \pi_i^c N$ and related to component m^c. Assuming the samples from different blocks to be independent, we have the likelihood of the model M_r as:

$$P(X|M_r) = \prod_{i=1}^{m^c} P(X_i|M_r). \tag{5.56}$$

We define now the binary variable $h_{ij} = h_i^c h_j^r$ that has value one if and only if the block X_i is assigned to the j^{th} component of M_r. If we assume also the samples correspond to each component of M_r are drawn independently, the likelihood of each block is given by:

$$P(X_i|M_r) = \sum_{j=1}^{m^r} \pi_j^r P(X_i|h_{ij} = 1, M_r) = \sum_{j=1}^{m^r} \pi_j^r \prod_{n=1}^{N_i} P(x_i^n|h_{ij} = 1, M_r). \tag{5.57}$$

The incomplete data likelihood under M_r for the whole N samples can be found by combining the equations 5.56 and 5.57 :

$$P(X|M_r) = \prod_{i=1}^{m^c} \sum_{j=1}^{m^r} \pi_j^r \prod_{n=1}^{N_i} P(x_i^n | h_{ij} = 1, M_r). \tag{5.58}$$

If we compare the above equation with EM-algorithm, we see that it is like the incomplete data likelihood in EM, but here we have one hidden variable for each of sample blocks and not for each sample. Thus, we obtain log-likelihood for complete data by:

$$\begin{aligned} \ln P(X, H|M_r) &= \ln \prod_{i=1}^{m^c} \prod_{j=1}^{m^r} [\pi_j^r P(X_i|h_{ij} = 1, M_r)]^{h_{ij}} && (5.59) \\ &= \sum_{i=1}^{m^c} \sum_{j=1}^{m^r} h_{ij} \ln(\pi_j^r P(X_i|h_{ij} = 1, M_r)) && (5.60) \end{aligned}$$

where H is a vector containing all the h_{ij}. Now, we can estimate the parameters of M_r based on EM algorithm, that leads us to the following E-step:

$$z_{ij} = \mathbb{E}[h_{ij}|X_i, M_r] = P(h_{ij} = 1|X_i, M_r) = \frac{P(X_i|h_{ij} = 1, M_r)\pi_j^r}{\sum_k P(X_i|h_{ij} = 1, M_r)\pi_k^r} \tag{5.61}$$

We take logarithm to the above equation in order to compute $P(X_i|h_{ij} = 1, M_r)$:

$$\ln P(X_i|h_{ij} = 1, M_r) = N_i[\frac{1}{N_i} \sum_{i=1}^{N_i} \ln P(x_i^n|h_{ij} = 1, M_r)] = N_i \mathbb{E}_{M_{c,i}}[\ln P(x|h_{ij} = 1, M_r)] \tag{5.62}$$

where $\mathbb{E}_{M_{c,i}}[x]$ is the expected value of x according to the i^{th} mixture components of M_c which is the component that X_i was drawn. For the Gaussian case it leads us to the following E-step:

$$z_{ij} = \frac{[\mathcal{N}(\mu_i^c, \mu_j^r, \Sigma_j^r) e^{-1/2tr((\Sigma_j^r)^{-1}\Sigma_i^c)}]^{N_i} \pi_j^r}{\sum_k [\mathcal{N}(\mu_i^c, \mu_k^r, \Sigma_k^r) e^{-1/2tr((\Sigma_k^r)^{-1}\Sigma_i^c)}]^{N_i} \pi_k^r} \tag{5.63}$$

To find M-step we should maximize

$$Q = \sum_{i=1}^{m^c} \sum_{j=1}^{m^r} z_{ij} \ln(\pi_j^r P(X_i|h_{ij} = 1, M_r)) \tag{5.64}$$

subject to the constraint $\sum_j \pi_j^r = 1$. For the Gaussian and having $N_i = \pi_i^c N$, it leads us to the following parameter update equations:

$$\pi_j^r = \frac{\sum_i z_{ij}}{m^c} \tag{5.65}$$

$$\mu_j^r = \frac{\sum_i z_{ij} \pi_i^c \mu_i^c}{\sum_i z_{ij} \pi_i^c} \tag{5.66}$$

$$\Sigma_j^r = \frac{\sum_i (z_{ij} \pi_i^c \Sigma_i^c + z_{ij} \pi_i^c (\mu_i^c - \mu_j^r)(\mu_i^c - \mu_j^r)^T)}{\sum_i z_{ij} \pi_i^c} \tag{5.67}$$

Thus, maximum likelihood estimate of M_r from X are expressed from the parameters of M_c using the equations 5.63 and 5.65 to 5.67, and hence no actual sampling from M_c needs to be done. But in this approach we need the number of components for finding the M_r and it is not defined automatically. In the next, we propose a Bayesian framework of our problem and then we use the grouping described here for a variational solution in section 5.4.3.

5.4 Variational Bayes-based aggregation for Gaussian mixtures

In chapter 3 we have talked about the idea to design an algorithm, in which the target model M_r, is to be estimated from the overcomplete model M_c rather than from data. An originality of our setting and solution arises from the need to estimate the parameters of the reduced model from an overcomplete model, rather than from data, the latter case being common setting for existing work in Variational Bayes-GMM.

In this section we try to drive an variational Bayestian framework for this problem. We propose to use the virtual sampling for reduction of GMM components discussed in 5.3 and then use variational Bayesian solution such as presented in section 5.2 to find an algorithm similar to variational-Bayes EM for estimating a GMM but operates on Gaussian components instead of data. In fact the aggregation is the same as in chapter 3, but we like to use a variational solution for the problem of reducing the components. The approach we propose consists in Bayesian modelling and estimation of the component grouping problem:

- Bayesian modelling automatically incorporates a criterion for correctly assessing the suitable model complexity,
- approximate inference through the variational approximation supplies reliable estimates for Θ^r at a computational cost similar to that of an EM algorithm. Further, component grouping and explicit cancelling of unnecessary components is carried out in a single process, i.e. scanning of all possible model complexities is avoided.

In the next, we first review concatenating the mixtures in section 5.4.1 In section 5.4.2 a Bayesian model of our problem will be introduced and finally in section 5.4.3 we apply a variational solution to the problem of the grouping of the components presented in section 5.3 to achieve the reduced Mixture from M_c.

5.4.1 Aggregation of the mixtures

As in chapter 3 , let two nodes each carry different probabilistic Gaussian mixture models, denoted $M_1(x)$ and $M_2(x)$, associated to the same multimedia class and hence hidden density $p(x)$. The mixtures can be expressed as $M_k(x) = \sum_{i=1}^{m_k} w_k^i N_k^i(x), \qquad k = 1, 2$ where $N_k^i(x)$ is a Gaussian component which mean is μ_k^i and covariance Σ_k^i and the w_k^i are scalar weights. Model M_k is estimated on a data set of size ,n_k located on node k. $p(x)$ can be estimated by concatenating incoming mixtures as

$$M_c(x) = \frac{1}{n_1 + n_2}\left[n_1 \sum_{i=1}^{m_1} w_1^i N_1^i(x) + n_2 \sum_{i=1}^{m_2} w_2^i N_2^i(x)\right] \tag{5.68}$$

Let M_c be parameterized by $\Theta^c = \{\pi^c, \mu^c, \Sigma^c\ \}$. Given M_c, we wish to infer the reduced mixture M_r, governed by $\Theta^r = \{\pi^r, \mu^r, \Sigma^r\}$. Implicit in this notation, the number of components m^r in M_r, should also be determined.

A possible direction to achieve mixture reduction consists in minimizing the Kullback divergence $KL(M_c \parallel M_r)$. However, the lack of closed-form in the case of divergence between two mixture models implies circumventing by approximations (e.g. [70, 186]). Further, since KL divergence relates directly to the (expected) log-likelihood, the essential issue of determining the appropriate model complexity (i.e. m^r) is left open.

5.4.2 Bayesian formulation of GMM reduction

As usual in mixture model problems, we cannot infer Θ^r directly from observations (here, Θ^c), but the introduction of latent variables is required. Denoted $Z = \{z_{ij}\}$, $1 \leq i \leq m^c$, $1 \leq j \leq m^r$, they are binary random variables indicating whether component i from M_c is assigned to component j in the reduced mixture M_r. We complete the specification of the model by defining a prior density on Θ^r, chosen as the most convenient conjugate priors :

- $p(\pi^r) = Dirichlet(\pi^r|\alpha)$. The Dirichlet distribution is a key point to the scheme we put forward, as its definition implements Occam's razor, i.e. it penalizes of an unnecessarily complex mixture. Indeed, choosing $\alpha < 1$ produces a density which value increases as more variables in π^r get closer to 0, i.e. when nonessential components vanish.
- $p(\mu^r, \Lambda^r) = p(\mu^r|\Lambda^r)p(\Lambda^r)$, a Gaussian-Wishart distribution.

The probabilistic model is completely described by the joint density $p(Z, \Theta^r, \Theta^c)$, and our goal is two fold :

- build the posterior $p(Z, \Theta^r|\Theta^c)$. In practice, Z_{ij} is entirely described by its expectation.
- find the model structure that maximizes the evidence $p(\Theta^c)$.

5.4.3 Variational resolution

Handling $p(Z, \Theta^r|\Theta^c)$ exactly is untractable, hence we resort to an approximate solution, through variational inference. In many circumstances, this method has shown to approach the quality of Monte-Carlo-based results, while preserving the moderate cost of low-order approximations of the posterior(eg. BIC). One can easily decompose the evidence $\ln p(X)$ into two terms, by introducing an intermediate distribution q:

$$\ln p(X) = F(q(Z, \Theta^r)) + KL(q \parallel p(Z, \Theta^r|X)) \tag{5.69}$$

where

$$F(q(Z, \Theta^r)) = \int q(Z, \Theta^r) \ln \frac{p(X, Z, \Theta^r)}{q(Z, \Theta^r)} \, dZ \, d\Theta^r \tag{5.70}$$

This decomposition shows that :

- $q(Z, \Theta^r)$ is an approximate density for the posterior $p(Z, \Theta^r|X)$;
- KL being always positive, $F(q(Z, \Theta^r))$ is a lower approximation for $\ln p(X)$.

Consequently, searching for $q(Z, \Theta^r)$ to match the following two objectives is in fact but one goal:

- finding $F(q(Z,\Theta^r))$ that best approximates $\ln p(X)$;
- finding $q(Z,\Theta^r)$ that best approximates $p(Z,\Theta^r|X)$.

The whole point of the above decomposition is that, if the choice for $q(Z,\Theta^r)$ is suitably constrained, optimizing $F(q(Z,\Theta^r))$ is tractable, whereas optimizing $\ln p(X)$ is not. An effective constraint consists in assuming that $q(Z,\Theta^r)$ can be decomposed as the product of independent factors, as expressed in equation 5.71.

$$p(Z,\Theta^r|\Theta^c) \approx q(Z,\pi^r,\mu^r,\Sigma^r) = q(Z)q(\pi^r)\prod_{k=1}^{m^r} q(\mu^r_k,\Sigma^r_k) \tag{5.71}$$

Optimizing $F(q(Z))$ can be conducted by optimizing for each random variable in turn, while averaging over other random variables (i.e. expectation, given a current estimate of the distribution for these other random variables). In the present case, this amounts to iterating between the two first following steps:

- Variational E-step : given an approximate posterior $q(\pi,\mu,\Sigma)$ (through its sufficient statistics), we approximate the posterior $q(Z)$ over the assignment variables Z, which is in fact entirely described by $< z_{ij} >$, where $< . >$ denotes expectation.

- Variational M-step : given an approximate posterior $q(Z)$, we compute an approximate posterior $q(\pi,\mu,\Sigma)$. Because we use conjugate priors, this steps amounts to updating model parameters. The analytical tractability appears by developing 5.70 and considering the expectation with respect to Z.

- Reduce the number of components in Mr, if appropriate : should any component k in M_r, have a low prior $\pi^r_k \approx 0$, it is deleted from the mixture. Let us recall that the Dirichlet prior tends to favor such configurations.

We detailed and derived these updates in section 5.2 in the case of estimation from data. Here we use a virtual sampling which discussed in section 5.3 to find E and M steps in our case. In fact substituting statistics from data by equations (5.63 to 5.67) supplies the actual variational E and M steps [27] which we detailed in the following:

We suppose a overcomplete Gaussian mixture model M_c with m^c components and parameters $\Theta^c = (\pi^c,\mu^c,\Lambda^c)$ and N virtual sample $X = \{X_1,\dots,X_{m^c}\}$ that each X_l is the samples with size $N_l = \pi^c N$ related to the lth component of the mixture. We first compute $\ln q^*(Z)$ which is

$$\ln q^*(Z) = \mathbb{E}_{\pi^r}[\ln p(Z|\pi^r)] + \mathbb{E}_{\mu^r,\Lambda^r}[p(X|Z,\mu^r,\Lambda^r)] \tag{5.72}$$

The first term can be calculated starting from the equation 5.21 as:

$$p(Z|\pi^r) = \prod_{n=1}^{N}\prod_{k=1}^{m^r}(\pi^r_k)^{z_nk} = \prod_{l=1}^{m^c}\prod_{k=1}^{m^r}(\pi^r_k)^{N_l z_{lk}} \tag{5.73}$$

because for all the point related to the component l, z_{nk} are the same and equal to z_{lk}. If we take logarithm we have

$$\ln p(Z|\pi^r) = \sum_{l=1}^{m^c}\sum_{k=1}^{m^r} N_l z_{lk} \ln \pi_k^r \tag{5.74}$$

For the second term, we calculate it from the equation 5.22:

$$p(X|Z,\mu^r,\Lambda^r) = \prod_{k=1}^{m^r}\prod_{n=1}^{N} \mathcal{N}(x_n|\mu_k^r,(\Lambda_k^r)^{-1})^{z_{nk}} = \prod_{k=1}^{m^r}\prod_{l=1}^{m^c}\left[\prod_{i=1}^{N_l}\mathcal{N}(x_{li}|\mu_k^r,(\Lambda_k^r)^{-1})\right]^{z_{lk}} \tag{5.75}$$

and after taking logarithm, we obtain:

$$\ln p(X|Z,\mu^r,\Lambda^r) = \sum_{k=1}^{m^r}\sum_{l=1}^{m^c} z_{lk}\sum_{i=1}^{N_l}[\ln \mathcal{N}(x_{li}|\mu_k^r,(\Lambda_k^{m^r})^{-1})] \tag{5.76}$$

Substituting the equations 5.74 and 5.76 in 5.72, give us :

$$\ln q^*(Z) = \sum_{l=1}^{m^c}\sum_{k=1}^{m^r} z_{lk}\ln\rho_{lk} + const \tag{5.77}$$

where we used the following equations (5.78 and 5.79) detailed in [27] that is achieved when N is sufficiently large:

$$\sum_{i=1}^{N_l}\ln\mathcal{N}(x_{li}|\mu_k^r,(\Lambda_k^r)^{-1}) \simeq N_l\frac{1}{2}[\ln|\Lambda_k^r| - Tr(\Lambda_k^r(\Lambda_l^c)^{-1})(\mu_l^c-\mu_k^r)^T\Lambda_k^r(\mu_l^c-\mu_k^r) - D\ln(2\pi)] \tag{5.78}$$

$$\begin{aligned}\ln(\rho_{lk}) = N\pi_k^c\Big\{&\mathbb{E}[\ln\pi_k^r] + \frac{1}{2}\mathbb{E}[\ln|\Lambda_k^r|] - \frac{1}{2}\ln(2\pi) - \frac{1}{2}\mathbb{E}_{\mu_k^r,\Lambda_k^r}[Tr(\Lambda_k^r(\Lambda_l^c)^{-1})\\ &+ (\mu_l^c-\mu_k^r)\Lambda_k^r(\mu_l^c-\mu_k^r)]\Big\}\end{aligned} \tag{5.79}$$

and the term $\mathbb{E}_{\mu_k^r,\Lambda_k^r}$ is computed to :

$$\mathbb{E}_{\mu_k^r,\Lambda_k^r}[Tr(\Lambda_k^r(\Lambda_l^c)^{-1}) + (\mu_l^c-\mu_k^r)\Lambda_k^r(\mu_l^c-\mu_k^r)] = D/\beta_k + \nu_k[Tr(W_k(\Lambda_l^c)^{-1}) + (\mu_l^c-m_k)^T W_K(\mu_l^c-m_k)] \tag{5.80}$$

Now for finding $q(Z)$ we take exponential from equation 5.77 thus we have:

$$q^*(Z) = \prod_{l=1}^{m^c}\prod_{k=1}^{m^r} r_{lk}^{z_{lk}} \propto \prod_{l=1}^{m^c}\prod_{k=1}^{m^r}\rho_{lk}^{z_{lk}} \tag{5.81}$$

where the quantities r_{lk} are the responsibilities which are normalized :

$$r_{lk} = \frac{\rho_{lk}}{\sum_{j=1}^{K}\rho_{lj}} \tag{5.82}$$

To compute the factor $q(\pi^r,\mu^r,\Lambda^r)$, we start the following equation as it is said in [19] for the real data:

$$\begin{aligned}\ln q^*(\pi^r, \mu^r, \Lambda^r) = \ln(\pi^r) + \sum_{k=1}^{m^r} \ln p(\mu_k^r, \Lambda_k^r) + \mathbb{E}_Z[\ln p(Z|\pi^r)] \\ + \sum_{k=1}^{m^r} \sum_{n=1}^{N} \mathbb{E}[Z_{nk}] \ln \mathcal{N}(x_n|\mu_k^r, (\Lambda_k^r)^{-1}) + const. \qquad (5.83)\end{aligned}$$

In order to drive the optimal solution for $q(\pi^r)$ we write only the terms that only depend on π^r and using equation 5.74 we obtain:

$$\begin{aligned}\ln q^*(\pi^r) &= \ln p(\pi^r) + \mathbb{E}_Z[\ln p(Z|\pi^r)] \\ &= (\alpha_0 - 1) \sum_{k=1}^{m^r} \ln \pi_k^r + \sum_{k=1}^{m^r} \sum_{l=1}^{m^c} N_l r_{lk} \ln \pi_k^r + const \qquad (5.84)\end{aligned}$$

which leads us to

$$q^*(\pi^r) = Dir(\pi^r|\alpha) \qquad (5.85)$$

where $\alpha = \{\alpha_1, \ldots, \alpha_{m^r}\}$ and

$$\alpha_k = \alpha_0 + \sum_{l=1}^{m^c} N_l r_{lk} \qquad (5.86)$$

Now we need to find the optimum variational posterior distribution over μ_k and Λ_k. For this mean, we start from 5.83 and find the terms which only depend μ_k or Λ_k and using the equation 5.78 we obtain:

$$\begin{aligned}\ln q^*(\mu_k^r, \Lambda_k^r) &= \ln \mathcal{N}(\mu_k^r|m_0, \beta_0 \Lambda_k^r) + \ln W(\Lambda_k^r|W_0, \nu_0) + \sum_{n=1}^{N} \mathbb{E}[Z_{nk}] \ln \mathcal{N}(x_n|\mu_k^r, \lambda_k^r) + const \\ &= \ln \mathcal{N}(\mu_k^r|m_0, \beta_0 \Lambda_k^r) + \ln W(\Lambda_k^r|W_0, \nu_0) + \sum_{l=1}^{m^c} \mathbb{E}[Z_{lk}] \sum_{i=1}^{N_l} \ln \mathcal{N}(x_{li}|\mu_k^r, \lambda_k^r) + const \\ &= -\frac{\beta_0}{2}(\mu_k^r - m_0)^T \Lambda_k (\mu_k^r - m_0) + \frac{1}{2} \ln |\Lambda_k^r| - \frac{1}{2} Tr(\Lambda_k^r W_0^{-1}) + \frac{(\nu_0 - D - 1)}{2} \ln |\Lambda_k^r| \\ &\quad - \frac{1}{2} \sum_{l=1}^{m_c} N_l r_{lk} [(\mu_l^c - \mu_k^r)^T \Lambda_k^r (\mu_l^c - \mu_k^r) - \ln |\Lambda_k^r| + Tr(\Lambda_k^r (\Lambda_l^c)^{-1}) + D \ln(2\pi)]\end{aligned} \qquad (5.87)$$

as we have $\ln q^*(\mu_k^r, \Lambda_k^r) = \ln q^*(\mu_k^r|\Lambda_k^r) + \ln q^*(\Lambda_k^r)$ (product rule of the probability), we need to compute the two factors $\ln q^*(\mu_k^r|\Lambda_k^r)$ and $\ln q^*(\Lambda_k^r)$. First we find the distribution for μ_k, so in equation 5.87 we consider only the terms which depend on μ_k and the result is:

$$\begin{aligned}
&\ln q^*(\mu_k^r|\Lambda_k^r) \\
&= -\frac{1}{2}(\mu_k^r)^T\left[\beta_0 + \sum_{l=1}^{m^c} N_l r_{lk}\right]\Lambda_k^r \mu_k^r + (\mu_k^r)^T\Lambda_k^r\left[\beta_0 m_0 + \sum_{l=1}^{m^c} N_l r_{lk}\mu_l^c\right] + const
\end{aligned} \tag{5.88}$$

which give us a Gaussian distribution for $\ln q^*(\mu_k^r|\Lambda_k^r)$:

$$q^*(\mu_k^r|\Lambda_k^r) = \mathcal{N}(\mu_k^r|m_k, \beta_k\Lambda_k^r) \tag{5.89}$$

where β_k and m_k are defined as follow:

$$\beta_k = \beta_0 + \sum_{l=1}^{m_c} N_l r_{lk} \tag{5.90}$$

$$m_k = \beta_0 m_0 + \sum_{l=1}^{m_c} N_l r_{lk}\mu_l^c \tag{5.91}$$

Now we use the equations 5.87 and 5.89 to find $q^*(\Lambda_k^r)$:

$$\begin{aligned}
\ln q^*(\Lambda_k^r) &= \ln q^*(\mu_k^r, \Lambda_k^r) - \ln q^*(\mu_k^r|\Lambda_k^r) \\
&= \Bigg[-\frac{\beta_0}{2}(\mu_k^r - m_0)^T\Lambda_k^r(\mu_k^r - m_0) + \frac{1}{2}\ln|\Lambda_k^r| - \frac{1}{2}Tr(\Lambda_k^r W_0^{-1}) + \frac{(\nu_0 - D - 1)}{2}\ln|\Lambda_k^r| \\
&\quad - \frac{1}{2}\sum_{l=1}^{m^c} N_l r_{lk}[(\mu_l^c - \mu_k^r)^T\Lambda_k^r(\mu_l^c - \mu_k^r) - \ln|\Lambda_k^r| + Tr(\Lambda_k^r(\Lambda_l^c)^{-1}) + D\ln(2\pi)]\Bigg] \\
&\quad - \left[-\frac{\beta_k}{2}(\mu_k^r - m_k)^T\Lambda_k^r(\mu_k^r - m_k) + \frac{1}{2}\ln|\Lambda_k^r|\right] + const.
\end{aligned} \tag{5.92}$$

As Λ_k^r is independent of μ_k^r, we remove the terms depend on μ_k^r then considering the terms depend on only on Λ_k^r and grouping the factors, we have :

$$\begin{aligned}
\ln q^*(\Lambda_k^r) &= \frac{1}{2}\left[(\nu_0 - D - 1) + \sum_{l=1}^{m^c} N_l r_{lk}\right]\ln|\Lambda_k^r| \\
&\quad + \frac{1}{2}\Bigg[-\beta_0(\mu_k^r - m_0)^T\Lambda_k^r(\mu_k^r - m_0) + \beta_k(\mu_k^r - m_k)^T\Lambda_k^r(\mu_k^r - m_k) - Tr(\Lambda_k^r W_0^{-1}) \\
&\quad - \sum_{l=1}^{m^c} N_l r_{lk}[(\mu_l^c - \mu_k^r)^T\Lambda_k^r(\mu_l^c - \mu_k^r) + Tr(\Lambda_k^r(\Lambda_l^c)^{-1})]\Bigg] \\
&= \frac{(\nu_k - D - 1)}{2}\ln|\Lambda_k^r| - \frac{1}{2}Tr(\Lambda_k^r W_k^{-1}) + const.
\end{aligned} \tag{5.93}$$

where W_k^{-1} and ν_k are defined as:

$$W_k^{-1} = W_0^{-1} + \beta_0 m_0 m_0^T - \beta_k m_k m_k^T + \sum_{l=1}^{m^c} N_l r_{lk} (\mu_l^c (\mu_l^c)^T + (\Lambda_l^c)^{-1}) \tag{5.94}$$

$$\nu_k = \nu_0 + \sum_{l=1}^{m^c} N_l r_{lk} \tag{5.95}$$

$$\tag{5.96}$$

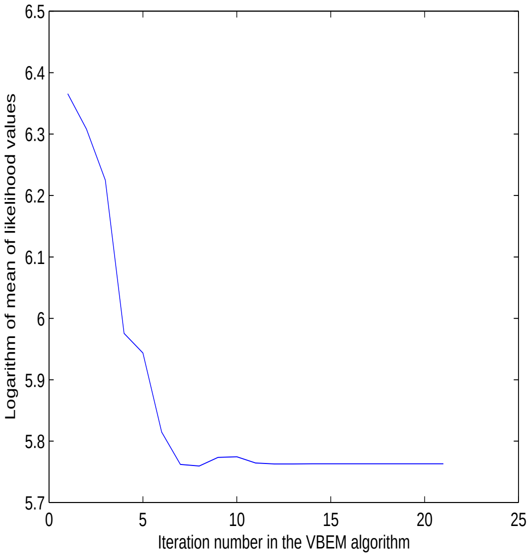
Thus, the optimization involves cycling between two step : VBE-step to evaluate r_{lk} (equation 5.82) and VBM-step to compute the variational parameters(the equations 5.86, 5.90, 5.91, 5.94 and 5.95).

5.5 Experimental results

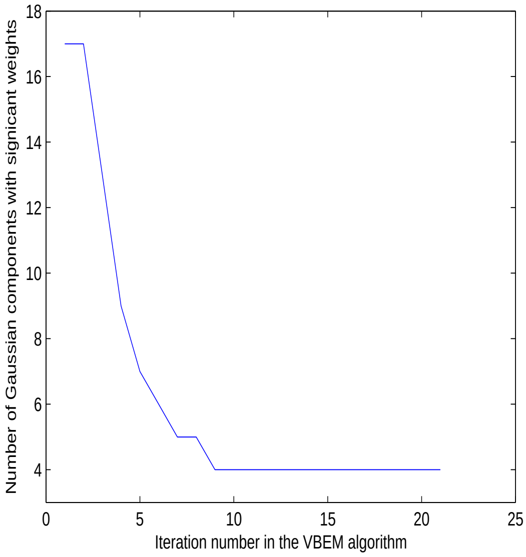
As in the previous chapters, we the example of distributed speaker recognition for experimental results, as it is a representative case where Gaussian mixtures are very popular. The technique however directly applies to a wide range of audiovisual classes. Each node owns different data from the same speaker and independently estimates its own model. Practically, EM with multiple initializations is employed in our experiments to alleviate the burden of local minima, but other techniques may be used. We evaluate the capability of each mixture on the network to model data D from the class of interest(here,a speaker) from $p(D)$, where D here is the union of the data dispatched over all nodes, which is never gathered when the practical system runs, but is a relevant figure of merit for external observation.

In first experiment we focus on an example mixture aggregation. The results of this example (figure 5.3) shows reducing the mixture created by concatenating two mixture models of two nodes using the modifies VBEM algorithm discussed in this chapter. Each of these two models was estimated from different data of the same speaker in a 2-D feature space. The figure 5.3(a) shows the evaluation of minus logarithm of mean likelihood to whole data D which should be minimized. A 6-component mixture is merged with an 11-component mixture. Starting from 17 components, the variational Bayes-EM technique jointly associates close-by Gaussian components and suppresses unnecessary ones, improving model evidence in the process. The figure 5.3(b) shows the progressive reduction of the number of meaningful Gaussian component from 17 to 4 as the iterative process set the proportion (weight) of some of them to zero.

To have a general result, we report here an experiment involving on 20 nodes, implemented in Matlab and we use the gossip protocol to concatenate two mixture in random in each step then find the reduced model using our algorithm. In our initialization we chose the initial number of components of new mixture (m^r) to the number of components of the concatenated mixture which usually decrease during the iterations. Figure 5.4 depicts, after each gossip cycle and on each node, the evolution of -log (mean-likelihood) to whole data. The figure 5.4(a) represent the average of log-likelihood values of all the mixtures. We observe that the evidence improves as the gossip progresses. The process stabilizes around a "collective model"(stabilization of evidence in fig 5.4(a) with very small variance). Figure 5.4(a) shows the average of number of components of all models in each step gossip. As this algorithm use only the few components of the model instead of huge multidimensional data points, it converge very fast comparing with standard VBEM algorithm.

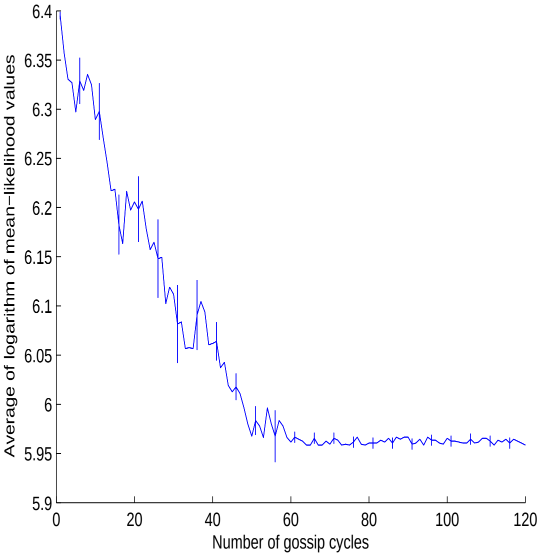


(a)

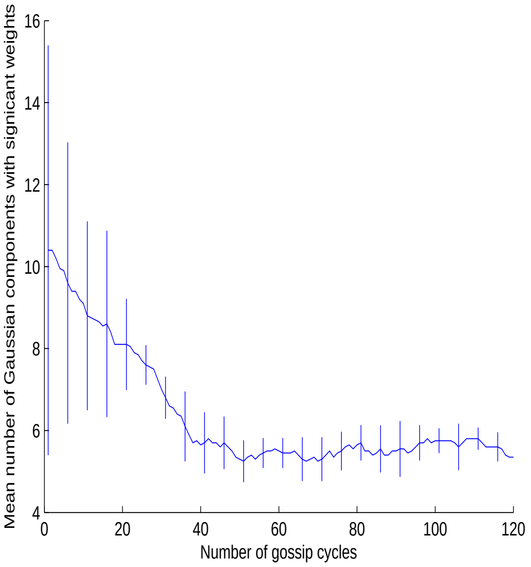


(b)

Figure 5.3 – An example of our VBEM algorithm for reduction the components of a concatenated model. a) -log(mean-likelihood) of the new model to whole data b) Number of components in the example of our VBEM algorithm for reduction the components of a concatenated model.



(a)



(b)

Figure 5.4 – Improving the evidence as the gossip progress. a) shows the average of -log(mean-likelihood) and variance of the new models to whole data b) Average number of components

5.6 Conclusion

This chapter discloses a scheme for distributed estimation of a Gaussian mixture model, in a gossip fashion. To this end, a powerful tool for estimating the parameters and the structure of such a model, namely variational-Bayes EM, is adapted to operate at mixture parameter-level rather than data-level, thus saving network resources. We demonstrate it on an average-size case, on acoustic data, yet the literature data shows that mixture models have wide applicability.

In comparison with the work presented in chapter 3 , the general context is the same, i.e. computing a GMM-based probabilistic model of a class, based on gossiping parameters. However, the present novel approach shifts the mixture aggregation principle significantly, in order to solve several drawbacks related to accuracy and speed of the previous technique :

- the approximation to the Kullback divergence between Gaussian mixtures is no longer required,
- the Bayesian criterion for assessing model complexity (i.e. how many Gaussians in the mixture) is more accurate (variational approximation instead of BIC), and it supplies a mechanism for automatically suppressing unnecessary components in the concatenated model (via the Dirichlet prior), hence saving the cost of systematically scanning all possible number of components, as in chapter 3.

CHAPTER 6

Conclusion

In this chapter, we summarize our main contributions. We also discuss some future directions of research for multimedia distributed learning.

6.1 Main contribution

This thesis addresses the problem of designing the distributed learning for multimedia recognition system in a decentralized fashion. In this work, the distribution of computation and data is due to the applicative context, in which independent systems cooperate. The algorithms and data from various contributors would cooperate towards a collective statistical learning. The key goal of the system is to obtain an estimate which quality is close to what would have been estimated in a centralized version. Despite its simplicity, this asynchronous, decentralized technique is very effective. The technical issue addressed by this thesis is the distributed computation of a probability density. To allow for a flexible evolution of the set of classes, we favored Gaussian mixture model that characterizes the class in feature space when new classes are introduced there is no need to update the description of old classes.

The good properties for our solution to the problem may summed up as follows :

- speed up by implementing coarse-grain parallelism over the set of nodes. This occurs at two levels :
  1. gossip-based parallelism of learning by merging
  2. for each step of gossip-based learning, parallelism in the computation of the likelihoods for validation
- robustness both in the distributing computing and statistical estimation senses, since:
  - any node may leave during the gossiping without causing major degradation or join and obtain, with high probability, an effective estimate of what has been previously collectively estimated on the network.
  - a very poor estimate in a minority of nodes does not affect the collectively estimated model.

Efficiency of the proposed technique comes the two main following features:

- merging density estimates between nodes only involves transmission of , and computation on mixture model parameters, rather than the generally large amount of multimedia data (or feature vectors that represent it). As a result:
  - the amount of information to be sent on the network is very low.
  - computation on nodes remains low, relatively to estimation tasks that operate on the multimedia data or feature vectors.

- during the gossip-based model learning phase, the complexity of any mixture (i.e. the number of Gaussian components) keeps a constant order of magnitude. Let us underline that the distributed learning phase and the querying phase, can fully overlap, since mixture reduction keeps the class representation directly ready for query evaluation.

Our main contributions are as follows:

• First, we presented an overview on machine learning techniques then multimedia indexing and retrieval techniques and distributed computing which are the elements of a distributed learning for multimedia recognition systems. Large amounts of labeled and unlabeled multimedia data (in the sense of supervised learning) is distributed on the Internet. Multimedia pattern recognition is generally a computation and data-intensive task. The perspective of exploiting collectively this data in a decentralized, distributed manner is made more realistic. Multimedia indexing is largely a learning/recognition issue and many softwares and different pattern recognition algorithms has been distributed on the Internet, but they work mainly locally and numerous pattern recognition algorithms are CPU demanding and the multimedia pattern to index is huge. We also took a look on distributed computing because of their good properties to deal with large amount of data learning and recognition systems distributed on a set of nodes. To yield general results, we strive to develop common algorithmic solutions with the right level of abstraction from the context. Thus, we assumed a peer to peer (P2P) distributed system architecture which is able to scale up to very large configurations. Then we present the gossip-based algorithms which can be used for computing the aggregations such as averages, sums and etc. These approaches mainly use data and need to transfer data on the network so they are not proper for multimedia data.

We then defined "distributed learning" which learns a model using the data distributed on the network without collecting the whole data on a central node or transferring data between the nodes. We presented a distributed learning technique for EM algorithm for distributed estimation of Gaussian mixtures.

For multimedia documents we need to find the approaches without transferring their huge data. We aim to have a fully decentralized algorithm that computes a global model for each class of data, with minimal processing and communication requirements and good fault-tolerant behavior. One way is distribution estimation of gaussian mixtures by aggregating the parameters of various models which is subject of this thesis.

• Second, we proposed a solution for distributed estimation of gaussian mixture by aggregating the mixture models which only requires moderate computation at each node and little data to transit between nodes. Models can be estimated locally, by a classical technique. We focused on aggregating local models to improve class characterization. The main feature of our approach is that participant nodes only transmit mixture model parameters rather than multimedia feature vectors, since aggregation only requires model parameters. Besides, an estimation of the appropriate number of components is carried out regularly, to enable scaling up. The proposed scheme aggregates local mixture model estimates by determining a suitable combination of their components. The mechanism for propagating mixtures between cooperating nodes that participate in the scheme is gossip. Any node may then supply, at any time, an estimate of the model which improves over time.

• Third, we addressed the problem of finding the best model in aggregation method and use the cross-validation method to improve the algorithm. Aggregating two mixture model does not guarantee that the resulting mixture model has better likelihood to the whole data than the two models. For improving the algorithm we used a combination of cross validation and merging to find the best model among many models which exist on the network for a unique class. The algorithm at first use a cross-validation manner and find a locally best model in a region according to the nodes in that region (neighborhoods of a node). After that it tries to find better models by merging the in that region models. This model can propagate to the network and compete to the other locally best models from other regions in the network.

• Fourth, we proposed an effective solution for aggregating the mixture models using variational Bayesian. Variational methods can be used for approximate the integrals required for Bayesian learning. The basic idea is to simultaneously approximate the distribution over both hidden states and parameters with a simpler distribution, usually by assuming the hidden states and parameters are independent. In this solution, aggregation is achieved through Bayesian modeling of the GMM component grouping problem and solved using a variational Bayes technique, applied at component level. This determines, through a single, low-cost yet accurate process, assignments of components that should be aggregated and the number of components in the mixture after aggregation.

6.2 Suggestion for future works

In this thesis, we proposed several techniques for low-cost distributed learning in a decentralized environment for a multimedia view point. We focused on the estimating a GMM model by merging many number of GMMs which each of them is already estimated by node on the network. But the general problem of estimating a model from distributed and decentralized data requires much more research. In this section there is a discussion on the open issues and future direction on this subject :

Using other mixtures instead of Gaussian : As we said already, our treatments have focussed on mixtures having Gaussian components, but Other mixtures also can be studied with attention to their properties. For example one of the Gaussian mixture models problem is their sensitivity to outliers which may lead to be intensively sensitive to small numbers of data points and thus cause the problem of over-estimates of the number of components. To overcome this problem Student-t distributions can be used which are more robust than Gaussians . [174] is a good illustration of using Student-t distribution in which they have develop a Bayesian approach to mixture modelling based on this distribution. Then they used the Student-t distribution as a marginalization over additional latent variables and derive a tractable variational inference algorithm for it, that Gaussian mixtures is a special case of this model.

Finding an optimal ranking algorithm : There are many mixtures on the network which we merged them using a gossip algorithm. We may find an algorithm to find the fastest convergence by ranking the edges (or nodes). These ranks determines which models are better to merge first for having the fastest convergence. In [23] and [22], they have studied distributed (gossip) algorithms, for exchanging information or for computing. Supposing an arbitrarily connected network of nodes, they analyzed the averaging problem under the gossip constraint, and found that the averaging time of a gossip algorithm depends on the second largest eigenvalue of a doubly stochastic matrix characterizing the algorithm. Based on this result, the fastest averaging algorithm of this class of algorithms, can be established using an optimization algorithm which assign the proportional weights to the edges to make the convergence as fast as possible. This framework is general and may be utilizable for our problem but by modifying to our settings in which the nodes have a GMM and in each gossip cycle a merging of two GMMs is occurred not an simple averaging of values.

Managing multiple classes simultaneously : In this thesis, we have proposed several approaches for low cost distributed learning of a unique class using Gaussian Mixture Model. Although this idea sounds interesting one of the difficulties is routing the multimedia queries toward the node that has the best model for that class when we have many classes on the network. For a file search, the query is broadcasted to the connecting nodes and each node propagate it to others while each nodes lookup its locally data and respond to the query. This is not a reliable way, because the node that has the best model may not received the request that increases the risque of misclassification while peers are forced to handle irrelevant query messages. There are also the solutions to solve the query broadcasting such as Chord[171] and CAN[147] but these approaches usually use distributed hash table (DHT) to map a filename to a key. DHT is not a good idea in our cases because while it needs a specific network structure, each node may estimate a different version of GMM that depend on its data and many other parameters and we have many model on the network for the same multimedia class. It seems that we need routing indices in each peer that route the query to the nodes that have the best models such as the approach that proposed in Crespo[43] for retrieving the text documents or in [97] for content-based image retrieval. This also reduce the network traffic. A simple solution may arise as follow : each peer which receives the query, try to find the nearest class for it according to its database, then the query will guide to the node which has the best model for that class according to its routing table.The question is that how the routing table can be created. A simple way may be to propagate the score of each local best model which found in the cross-validation manner (see chapter 4) and each node that receive this score may create or modify the field related to that class.

List of publications

Journal

A. Nikseresht and M. Gelgon, Gossip-based computation of a Gaussian mixture model for distributed multimedia indexing, *IEEE Transactions on Multimedia*, Volume 10, Issue 3, Pages 385-392, 2008.

International conferences

M. Gelgon, A. Nikseresht, Decentralized learning of a Gaussian Mixture with variational Bayes-based aggregation, *Euromicro conference on Parallel and Distributed Processing (PDP'2008)*, Toulouse, France, February 2008.

A. Nikseresht, M. Gelgon, An approach to distributed model selection for multimedia content-based indexing in a decentralized context, *IEEE International Conference on Digital Information Management (ICDIM'2007)*, Lyon, France, October 2007.

A. Pigeau, A. Nikseresht, M. Gelgon, Fast tracking of hierarchical partitions with approximate KL-divergence for geo-temporal organization of personal images, *The 22nd Annual ACM Symposium on Applied Computing (SAC'2007)* ,COEX Convention Center, Seoul, Korea, March 2007.

A. Nikseresht, M. Gelgon, Decentralized Distributed Learning of a Multimedia Class for Content-based Indexing, *14th Euromicro conference on Parallel, Distributed and network-based Processing (PDP'2006)*,Montbeliard, France, February 2006.

A. Nikseresht, M. Gelgon, Low-cost Distributed Learning of a Gaussian Mixture Model for Multimedia Content-based Indexing on a P2P Network, *7th ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR'2005)*, Singapore, November 2005. This paper obtained a student grant award from IBM/Microsoft.

National conferences

M. Gelgon, A. Pigeau, A. Nikseresht, Suivi de partitions géo-temporelles à partir d'une divergence de Kullback-Leibler modifiée en vue de la navigation dans une collection d'images personnelles, *Journées Compression et Représentation des Signaux Audiovisuels (CORESA'2006)*, Caen, France, November 2006.

A. Nikseresht, M. Gelgon, Agrégation légère de mélange de lois gaussiennes en vue de l'indexation multimédia répartie sur un réseau pair à pair, *Congrès Reconnaissance des Formes et Intelligence Artificielle (RFIA'2006)*, Tours, France, January 2006.

A. Nikseresht, M. Gelgon, Agrégation légère de mélange de lois gaussiennes en vue de l'indexation multimédia répartie, *Journées Compression et Representation des Signaux Audiovisuelles (CORESA'2005)*, Rennes, France, November 2005.

Bibliography

[1] D. Agrawal, A. El. Abbadi, and R.C. Steinke. Algorithms in replicated databases. *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 161–172, 1997.

[2] H. Akaike. Information theory as an extension of the maximum likelihood principle. *Second International Symposium on Information Theory*, pages 267–281, 1973.

[3] H. Akaike. A new look at the statistical model identification problem. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[4] H. Akaike. Information measures and model selection. *Bulletin of the International Statistical Institute*, 50:277–290, 1983.

[5] E. Ardizzone and M. La Cascia. Automatic video database indexing and retrieval. *Multimedia Tools and Applications*, 4:29–56, 1997.

[6] J. Assfalg, A. Del Bimbo, and P. Pala. Three-dimensional interfaces for querying by example in content-based image retrieval. *IEEE Trans. Visualization and Computer Graphics*, 8(4):305–318, 2002.

[7] H. Attias. Inferring parameters and structure of latent variable models by variational bayes. *Proceedings of Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 21–30, 1999.

[8] H. Attias. A variational bayesian framework for graphical models. *Neural Information Processing System (NIPS) Conference*, November 1999.

[9] H. Attias. Learning in high dimensions: Modular mixture models. *Proc. AI and statistics*, 2001.

[10] J.D. Banfield and A.E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49:803–821, 1993.

[11] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Stanford University, 2003.

[12] R. Bayer. Symmetric binary b-trees: Data structure and maintenance algorithms. *Acta Informatica*, 1:290–306, 1972.

[13] M.J. Beal. Variational algorithms for approximate bayesian inference. *PhD thesis, Gatsby Computational Neuroscience Unit, University College London*, 2003.

[14] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 322–331, Atlantic City, USA, 1990.

[15] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18:509–517, 1975.

[16] C. Biernacki, G. Celeux, and G. Govaert. Choosing starting values for the em algorithm for getting the highest likelihood in multivariate gaussian mixture models. *Computational Statistics and Data Analysis*, 41:561–575, 2003.

[17] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM TOCS*, 17:41–88, 1999.

[18] M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[19] M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[20] A. Blum and T.Mitchell. Combining labeled and unlabeled data with co-training. *COLT: Proceedings of the Workshop on Computational Learning Theory,Morgan Kaufmann*, pages 92–100, 1998.

[21] P. Bouthemy, M. Gelgon, and F. Ganansia. A unified approach to shot change detection and camera motion characterization. *IEEE Trans. Circuits and Systems for Video Technology*, 9(7):1030–1044, Jul 1999.

[22] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: design, analysis and applications. In *to appear at IEEE INFOCOM'2005*, Miami, March 2005.

[23] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

[24] H. Bozdogan. *Multi-Sample Cluster Analysis and Approaches to Validity Studies in Clustering Individuals.* PhD thesis, Department of Mathematics, University of Illinois, Chicago, 1981.

[25] H. Bozdogan. Determining the number of clusters in the standard multivariate normal mixture model using model selection criteria. Technical report, Department of Mathematics, University of Illinois, Chicago, June 1983.

[26] L. Breiman, J.H. Friedman, R.A. Olshen, and P.J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[27] Pierrick Bruneau, Marc Gelgon, and Fabien Picarougne. Parameter-based reduction of gaussian mixture models with a variational-bayes approach. In *Proc. International Conference on Pattern Recognition (ICPR'08)*, Tampa (Florida), USA, December 2008.

[28] E. Bruno and D. Pellerin. Video structuring, indexing and retrieval based on global motion wavelet coefficients. In *6th Int. Conf. Pattern Recognition*, volume 3, pages 287–290, Quebec City, Canada, Aug 2002.

[29] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream: High-bandwidth multicast in cooperative environment. In *ACM Symposium on Operating Systems Principles*, New York, Oct 2003.

[30] C. Celeux and J. Dielbolt. The sem algorithm: a probabilistic teacher algorithm derived from the em algorithm for the mixture problem. *Computational Statistics Quarterly*, 2(1):73–92, 1985.

[31] G. Celeux and G. Govaert. Gaussian parsimonious models. *Pattern Recognition*, 28(5):781–793, may 1995.

[32] A. Chalechale, G. Naghdy, and A. Mertins. Sketch-based image matching using angular partitioning. *IEEE Trans. Systems, Man, and Cybernetics*, 35(1):28–41, 2005.

[33] Y. Chan and S.Y. Kung. A hierarchical algorithm for image retrieval by sketch. *First IEEE Workshop on Multimedia Signal Processing*, pages 564–569, 1997.

[34] E.Y. Chang, K. Goh, G. Sychay, and G. Wu. Cbsa: Content-based soft annotation for multimodal image retrieval using bayes point machines. *IEEE Trans. Circuits and Systems for Video Technology*, 13(1):26–38, 2003.

[35] O. Chapelle, B. Schölkopf, and A. Zien. Semi-supervised learning. *MIT Press, Cambridge, MA*, 2006.

[36] Y. Chen, J.Z. Wang, and R. Krovetz. Clue: Cluster-based retrieval of images by unsupervised learning. *IEEE Trans. Image Processing*, 14(8):1187–1201, 2005.

[37] D.M. Chickering and D. Heckerman. Fast learning from sparse data. *Proc. NIPS-99*, 1999.

[38] E. Cohen, A. Fiat, and H. Kaplan. Associative search in peer to peer networks: Harnessing latent semantics. *IEEE INFOCOM'03*, April 2003.

[39] D. Comer. The ubiquitous b-tree. *ACM Computing Surveys*, 11:121–137, 1979.

[40] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3), Septembre 1995.

[41] I.J. Cox, M.L. Miller, T.P. Minka, T.V. Papathomas, and P.N. Yianilos. The bayesian image retrieval system, pichunter: Theory, implementation, and psychophysical experiments. *IEEE Trans. Image Processing*, 9(1):20–37, 2000.

[42] F. Cozman, M. Cirelo, T.S. Huang, I. Cohen, and N. Sebe. Semisupervised learning of classifiers : theory, algorithms and their applications to human-computer interaction. *IEEE Trans. on pattern Analysis and Machine Intelligence*, 26(12):1553–1567, Dec 2004.

[43] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. *Proceedings of the 28th Conference on Distributed Computing Systems*, July 2002.

[44] A. Csillaghy, H. Hinterberger, and A. Benz. Content based image retrieval in astronomy. *Information Retrieval*, 3(3):229–241, 2000.

[45] R. Datta, D. Joshi, J. Li, and J.Z. Wang. Studying aesthetics in photographic images using a computational approach. In *Proc. 9th European Conference on Computer Vision (ECCV)*, Graz, Austria, May 2006.

[46] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing*, pages 1–12, 1987.

[47] A.P. Dempster, N.M. Laid, and D.B. Rubin. Maximum likelihood for incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.

[48] Y. Deng and B. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(8):800–810, 2001.

[49] C. Djeraba. Content-based multimedia indexing and retrieval. *IEEE MultiMedia*, 9(2):18–22, 2002.

[50] C. Djeraba. Association and content-based retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):118–135, 2003.

[51] M.N. Do and M. Vetterli. Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance. *IEEE Trans. Image Processing*, 11(2):146–158, 2002.

[52] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003.

[53] Richard O. Duda and Peter E. Hart. *Pattern classification and scene analysis.* Wiley-Interscience, 1973.

[54] B. Efron and R.J. Tibshirani. *an Introduction to the Bootstrap.* Chapman and Hall/CRC, 1993.

[55] P.T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. From epidemics to distributed computing. *IEEE Computer*, 37(5):60–67, 2004.

[56] R. Fablet, P. Bouthemy, and P. Perez. Non parametric motion characterization using causal probabilistic models for video indexing and retrieval. *IEEE trans. on Image Processing*, 11(4):393–407, April 2001.

[57] O. Faugeras. Three-dimensional computer vision, a geometric viewpoint. *Cambridge, MA: MIT Press*, 1993.

[58] D. Feng, W.C. Siu, and H.J. Zhang. *Multimedia information retrieval and management: Technological fundamentals and applications.* Springer, Berlin, 2003.

[59] R.A. Finkel and J.L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.

[60] R.A. Fisher. *Statistical methods and scientific inferences.* Edinburgh : Oliver and Boyd, 1956.

[61] R. Fletcher. *Practical methods of optimization.* Whiley, second edition, 1987.

[62] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, B. Qian Huang Dom, M. Gorkani, J. Hafner, and D. Lee. Query by image and video content: the qbic system. *IEEE Computer*, 28(9):23–32, 1995.

[63] M.J. Fonseca and J.A. Jorge. Indexing high-dimensional data for content-based retrieval in large database. In *Proceedings of the Eighth International Conference on Database Systems for Advanced Applications*, Kyoto, Japan, 2003.

[64] G. Forman and B. Zhang. Distributed data clustering can be efficient and exact. *ACM SIGKDD Explorations*, 2(2):34–38, 2000.

[65] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

[66] W.T. Fu and J.R. Anderson. From recurrent choice to skill learning: A reinforcement-learning model. *Journal of Experimental Psychology: General*, 135(2):184–206, 2006.

[67] D. Gavrila. The visual analysis of human movement: a survey. *Comput. Vis. Image Understand*, 73(1):82–98, 1999.

[68] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis.* Chapman and Hall, London, 1998.

[69] Gnutella. At http://gnutella.wego.com.

[70] Jacob Goldberger and Sam Roweis. Hierarchical clustering of a mixture model. In *Proc. of Neural Information Processing Systems 17 (NIPS'04)*, pages 505–512, December 2004.

[71] Y.H. Gong, C.H. Chuan, and X.Y. Guo. Image indexing and retrieval based on color histograms. *Multimedia Tools and Applications*, 2:133–156, 1996.

[72] A.A. Goodrum. Image information retrieval: An overview of current research. *Informing Science, Special Issue on Information Sciences*, 3(2), 2000.

[73] P.J. Green. On the use of the EM algorithm for penalized likelihood estimation. *Journal of the Royal Statistical Society B*, 52(3):443–452, 1990.

[74] H. Greenspan, J. Goldberger, and A. Mayer. Probabilistic space-time video modeling via piecewise gmm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):384–396, March 2004.

[75] L. Guan, S.Y. Kung, and J. Larsen. *Multimedia image and video processing.* CRC Press, Newyork, 2001.

[76] V.N. Gudivada and V.V. Raghavan. Content-based image retrieval systems. *IEEE Computer*, 28(9):18–22, 1995.

[77] I. Gupta, R. van Renesse, and K. Birman. Scalable fault-tolerant aggregation in large process groups. *Proc. Conf. on Dependable Systems and Networks*, pages 433–442, 2001.

[78] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston,USA, 1984.

[79] R. Guy, G. Popek, and T.w.jr. Page. Consistency algorithms for optimistic replication. *Proceedings of the First International Conference on Network Protocols, IEEE*, 1993.

[80] R. Hammoud and R. Mohr. Gaussian mixture densities for video object recognition. In *Proc. on Int. Conf. on Pattern Recognition (ICPR'2000)*, pages 71–75, Barcelona, Spain, August 2000.

[81] D. Heckerman. A tutorial on learning with bayesian networks. *Learning in graphical models*, 1999.

[82] G.E. Hinton and T.J. Sejnowski. Unsupervised learning and map formation: Foundations of neural computation. *MIT Press*, 1999.

[83] J. Huang, S. R. Kumar, M. Mitra, W.J. Zhu, and R. Zabih. Image indexing using color correlograms. *Proc. of the IEEE Computer Vision And Pattern Recognition*, pages 762–768, 1997.

[84] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[85] A. Jain and D. Zongker. Feature selection: Evaluation, application and small sample performance. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(2):153–158, 1997.

[86] M. Jelasity, W. Kowalczyk, and M. van Steen. Newscast computing. *Technical report, Dept. of Computer Science, Vrije Universiteit Amsterdam,IR-CS-006*, 2003.

[87] H. Jiang and S. Jin. Scalable and robust aggregation techniques for extracting statistical information in sensor networks. *26th IEEE International Conference on Distributed Computing Systems (ICDCS06)*, 2006.

[88] R. Jin and A. Hauptmann. Using a probabilistic source model for comparing images. *Proc. IEEE ICIP*, 2002.

[89] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Machin Learning*, 37(2):183–233, 1999.

[90] M.I. Jordan and R.A. Jacobs. Hierarchies of adaptive experts. *Advances in Neural Information Processing Systems*, 4:985–992, 1992.

[91] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, March 1994.

[92] L.P. Kaelbling, M.L. Littman, and A.W.Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[93] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. *Proc. 41st IEEE FOCS*, pages 565–574, 2000.

[94] R.E. Kass and A.E. Raftery. Bayes factors and model uncertainty. *Journal of the American Statistical Association*, 90:773–795, 1995.

[95] T. Kato. Database architecture for content-based image retrieval. *Image Storage and Retrieval Systems , Proc SPIE*, 1662:112–123, 1992.

[96] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 482–491, 2003.

[97] Irwin King, Cheuk Hang Ng, and Ka Cheung Sia. Distributed content-based visual information retrieval system on peer-to-peer networks. *ACM Transactions on Information Systems (TOIS)*, 22:477–501, July 2004.

[98] Ron 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.

[99] W. Kowalczyk and N. Vlassis. Newscast EM. In MIT Press, editor, *Proc. of Neural Information Processing Systems (NIPS) 17*, 2005.

[100] A.K. Kurtz. research test of rorschach test. *Personnel Psychology*, 1:41–53, 1948.

[101] D. Lee, R. Barber, W. Niblack, M. Flickner, J. Hafner, and D.Petkovic. Query by image content using multiple objects and multiple features: user interface issues. In *Proceedings of International Conference on Image Processing (ICIP)*, pages 76–80, Austin, Texas, 1994.

[102] E. Levine and E. Domany. Resampling method for unsupervised estimation of cluster validity. *Neural Computation*, 13(11):2573–2593, November 2001.

[103] J. Li, B.T. Loo, J. Hellerstein, F. Kaashoek, D.R. Karger, and R. Morris. The feasibility of peer-to-peer web indexing and search. *IPTPS'03*, February 2003.

[104] J. Li and J.Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(9):1075–1088, 2003.

[105] J. Li and J.Z. Wang. Studying digital imagery of ancient paintings by mixtures of stochastic models. *IEEE Trans. Image Processing*, 13(3):340–353, 2004.

[106] J. Li and J.Z. Wang. Alip: the automatic linguistic indexing of pictures system. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:1208–1209, June 2005.

[107] J.Q. Li and A.R. Barron. Mixture density estimation. *Advances in Neural Information Processing Systems 12*, 2000.

[108] H. Linhart and W. Zucchini. *Model Selection.* John Wiley & Sons Inc, New York, 1986.

[109] S.P. Lloyd. Least squares quantization in pcm. *IEEE transactions on Information Theory*, 28(2):129–137, March 1982.

[110] M. Losada and E. Heaphy. The role of positivity and connectivity in the performance of business teams: A nonlinear dynamics model. *American Behavioral Scientist*, 47(6):740–765, 2004.

[111] C.E. Lunneborg. *Data Analysis by Resampling: Concepts and Applications.* Duxbury Press, 1st edition, December 1999.

[112] A. Kononov M. Rabinovich, N. Gehani. Scalable update propagation in epidemic replicated databases. *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, 1057:207–222, 1996.

[113] W.Y. Ma and B.S. Manjunath. Texture-based pattern retrieval from image databases. *Multimedia Tools and Applications*, 2:35–51, 1996.

[114] D. MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

[115] J. MacQueen. Some methods for classification and analysis of multivariate observation. *proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967.

[116] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. *Proc. 5th Symp. on Operating Systems Design and Implementation*, 2002.

[117] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:837–842, 1996.

[118] J.R. Mathiassen, A. Skavhaug, and K. Bo. Texture similarity measure using kullback-leibler divergence between gamma distributions. *Proc. ECCV*, 2002.

[119] G.J. McLachlan and D. Peel. *Finite Mixture Models.* John Wiley & Sons, 2000.

[120] M. Meila. *Learning with mixures of trees.* PhD thesis, MIT, 1998.

[121] M. Meila and D. Heckerman. An experimental comparison of several clustering and initialization methods. In *Proc. 14th Ann. Conf. on Uncertainty in Artificial Intelligence*, pages 386–395, San Francisco, 1998.

[122] R. Milanese, D. Squire, and T. Pun. Correspondence analysis and hierarchical indexing for content-based image retrieval. In *International Conference on Image Processing (ICIP'96)*, volume 3, pages 859–862, Lausanne, Switzerland, September 1996.

[123] D. Milojicic, V. Kalogeraki, R. Lukose, L. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to peer computing. Technical Report 57, Hewlett-Packard, 2002.

[124] B. Minaei-Bidgoli, A. Topchy, and W.F. Punch. Ensembles of partitions via data resampling. In *International Conference on Information Technology : Coding and Computing (ITCC)*, pages 188–192, Las Vegas, April 2004.

[125] A. Moore. Very fast em-based mixture model clustering using multiresolution kd-trees. *Proc. NIPS-98*, 1999.

[126] W.T. Muller, M. Eisenhardt, and A. Henrich. Efficient content-based p2p image retrieval using peer content descriptions. In *Proc. of SPIE Internet Imaging V*, volume 5304, pages 57–68, December 2003.

[127] M. Naaman, Y. J. Song, A. Paepcke, and H. Garcia-Molina. Automatic organization for digital photographs with geographic coordinates. *Proc.of ACM/IEEE Conference on Digital libraries (JCDL'2004)*, pages 53–62, 2004.

[128] Krishna Nadiminti, Marcos Dias de Assunção, and Rajkumar Buyya. Distributed systems and recent innovations: Challenges and benefits. *InfoNet Magazine*, 16(3), September 2006.

[129] Napster. At http://www.napster.com.

[130] A. Natsev, R. Rastogi, and K. Shim. Walrus: A similarity retrieval algorithm for image databases. *IEEE Trans. Knowledge and Data Engineering*, 16(3):301–316, 2004.

[131] R.M. Neal. Probabilistic inference using markov chain monte carlo methods. *Rapport technique CRG-TR-93-1, Department of Computer Science, University of Toronto*, 1993.

[132] R.M. Neal and G.E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 15(2):355–368, 1998.

[133] Chong-Wah Ngo, Ting-Chuen Pong, and Hong-Jiang Zhang. On clustering and retrieval of video shots through temporal slices analysis. *IEEE Trans. Multimedia*, 4(4):446–458, Aug 2002.

[134] J. Nocedal and S.J. Wright. *Numerical optimization.* Springer, 1999.

[135] R. D. Nowak. Distributed em algorithms for density estimation and clustering in sensor networks. *IEEE Trans. on Signal Processing*, 51(8):2245–2253, 2003.

[136] J.R.J. Nunnink, J.J. Verbeek, and N. Vlassis. Accelerated greedy mixture learning. In *Proc. Belgian-Dutch Conference on Machine Learning*, Belgium, January 2004.

[137] OpenSourceCommunity. The free network project. in http://freenet.sourceforge.net/. 2001.

[138] D. Ormoneit and V. Tresp. Averaging, maximum penalized likelihood and bayesian estimation for improving gaussian mixture probability density estimates. *IEEE Transactions on Neural Networks*, 9(4):639–649, 1998.

[139] T.H. Painter, J. Dozier, D.A. Roberts, R.E. Davis, and R.O. Green. Retrieval of subpixel snow-covered area and grain size from imaging spectrometer data. *Remote Sensing of Environment*, 85(1):64–77, 2003.

[140] E.G.M. Petrakis, C. Faloutsos, and K.I. Lin. Imagemap: An image indexing method based on spatial similarity. *IEEE Trans. Knowledge and Data Engineering*, 14(5):979–987, 2002.

[141] A. Pigeau and M. Gelgon. Building and tracking hierarchical partitions of image collections on mobile devices. In *ACM Multimedia conference*, pages 141–151, Singapore, November 2005.

[142] G. Piriou. Modélisation statistique du mouvement dans des séquences d'images pour la reconnaissance de contenus dynamiques. *Thèse de l'Université de Rennes 1*, Dec 2005.

[143] G. Piriou, P. Bouthemy, and J.-F. Yao. Recognition of dynamic video contents with global probabilistic models of visual motion. *IEEE Trans. on Image Processing*, 15(11):3417–3430, 2006.

[144] J. Ponce, M. Hebed, C. schmid, and A. Zisserman. *Towards category-level object recognition.* Springer, 2006.

[145] R. Price, T. Chua, and S. Al-Hawamde. Applying relevance feedback to a photo archival system. *Journal of Information Science*, 18(3):203–215, 1992.

[146] M. Quenouille. Approximate tests of correlation in time series. *Journal of the Royal Statistical Society*, 11:18–84, 1949.

[147] Sylvia Ratnasamy, Paul Francisan, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. *Proceedings of ACM SIGCOMM*, pages 161–172, 2001.

[148] D. Reynolds. Speaker identification and verification using gaussian speaker models. *Speech communication*, 17:91–108, 1995.

[149] D.A. Reynolds. A gaussian mixture modeling approach to text-independent speaker identification. *Ph.D. thesis, Georgia Institute of Technology*, September 1992.

[150] D.A. Reynolds. Automatic speaker recognition using gaussian mixture speaker models. *Lincoln Lab. J.*, 8:173–192, 1996.

[151] D.A. Reynolds and R.C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Trans. Speech Audio Process*, 3:72–83, 1995.

[152] S. Rhea and J. Kubiatowicz. Probabilistic location and routing. *IEEE INFOCOM'02*, June 2002.

[153] J. Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific, 1989.

[154] J.J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART retrieval system - Experiments in automatic document processing*, pages 313–323, Englewood Cliffs, NJ,USA, 1971. Prentice-Hall.

[155] K. Rodden and K. Wood. How do people manage their digital photographs? *ACM Conference on Human Factors in Computing Systems*, pages 409–416, 2003.

[156] R.C. Rose and D.A. Reynolds. Text-independent speaker identification using automatic acoustic segmentation. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 293–296, 1990.

[157] Y. Rubner, C. Tomasi, and L.J. Guibas. A metric for distribution with applications to image databases. *Proc. IEEE ICCV*, 2002.

[158] Y. Rui and T. Huang. Learning based relevance feedback in image retrieval. *Advances in image processing and understanding: A festschrift for Thomas S. Huang*, pages 163–182, 2002.

[159] Y. Rui, T.S. Huang, and S. Mehrotra. Relevance feedback techniques in interactive content-based image retrieval. *Storage and Retrieval for Image and Video Databases VI, Proc SPIE*, 3312:25–36, 1997.

[160] G. Salton. *The SMART retrieval system - experiments in automatic document processing.* Prentice-Hall, New Jersey, 1971.

[161] S. Salvador and P. Chan. Determining the number of clusters/segments in hierarchical clustering/ segmentation algorithms. *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 576–584, November 2004.

[162] S. Satoh, Y. Nakamura, and T. Kanade. Name-it : naming and detecting faces in news videos. *IEEE Multimedia*, 6(1):22–35, Jan-Mar 1999.

[163] C. Schmid. Weakly supervised learning of visual models and its application to content-based retrieval. *International Journal of Computer Vision*, 56(1):7–16, 2004.

[164] M. Schroder, H. Rehrauer, K. Seidel, and M. Datcu. Interactive learning and probabilistic retrieval in remote sensing image archives. *IEEE Trans. Geoscience and Remote Sensing*, 38(5):2288–2298, 2000.

[165] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[166] S. Sclaroff, L. Taycher, and M. LaCascia. Imagerover: a content-based image browser for the world-wide web. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries*, San Juan, Puerto Rico, June 1997.

[167] B. Shah, V. Raghavan, and Dhatric. Efficient and effective content-based image retrieval using space transformation. In *Proceedings of the 10th International Multimedia Modelling Conference*, Brisbane, Australia, 2004.

[168] J. Shen, J. Shepherd, and A.H.H. Ngu. Towards effective content-based music retrieval with multiple acoustic feature combination. *IEEE Transactions on Multimedia*, 8(6):1179–1189, 2006.

[169] J.R. Smith and S.F. Chang. An image and video search engine for the world-wide web. *Storage and Retrieval for Image and Video Databases V, Proc SPIE*, 3022:84–95, 1997.

[170] P. Smyth. Clustering using monte-carlo cross-validation. In *In proc, the second International Conference on Knowledge Discovery and Data Mining*, pages 126–133, Portland, August 1996.

[171] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet application. *IEEE/ACM Transactions on Networking*, 11(1):17–32, February 2003.

[172] Z. Su, H.J. Zhang, S. Li, and S. Ma. Relevance feedback in content-based image retrieval: Bayesian framework, feature subspaces, and progressive learning. *IEEE Trans. Image Processing*, 12(8):924–937, 2003.

[173] R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *MIT Press. ISBN 0-262-19398-1*, 1998.

[174] M. Svensen and C.M. Bishop. Robust bayesian mixture modelling. *Neurocomputing*, 64:235–252, March 2005.

[175] C. Tang, S. Dwarkadas, and Z. Xu. On scaling latent semantic indexing on large peer-to-peer systems. In *Proc. of ACM SIGIR (SIG on Information Retrieval*, pages 145–153, Sheffield, U.K., July 2004.

[176] R. Tibshirani, G. Walther, D. Botstein, and P. Brown. Cluster validation by prediction strength. *Rapport technique, Statistics Department, Stanford University*, 2001.

[177] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. *Journal of the Royal Statistical Society: : Series B (Statistical Methodology)*, 63:411–423, 2001.

[178] M.E. Tipping. The relevance vector machine. *Advances in Neural Information Processing Systems*, 12:652–658, 2000.

[179] F. Tomita and S. Tsuji. *Computer analysis of visual textures.* Kluwer Academic Publishers, Boston, 1990.

[180] N. Ueda, R. Nakano, Z. Ghahramani, and G.E. Hinton. Smem algorithm for mixture models. *Neural Computation 12*, pages 2109–2128, 2000.

[181] R. van Renesse, K. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM TOCS*, 2003.

[182] V. N. Vapnik. Statistical learning theory. *Wiley, New York, ISBN: 978-0-471-03003-4*, pages 339–371, 1998.

[183] N. Vasconcelos. Image indexing with mixture hierarchies. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, pages 3–10, Kauai,USA, December 2001.

[184] N. Vasconcelos. On the efficient evaluation of probabilistic similarity functions for image retrieval. *IEEE Trans. Information Theory*, 50(7):1482–1496, 2004.

[185] N. Vasconcelos and A.Lippman. Feature representations for image retrieval: Beyond the color histogram. In *Proc. of the Int. Conference on Multimedia and Expo*, New York, August 2000.

[186] N. Vasconcelos and A. Lippman. Learning mixture hierarchies. *Neural Information Processing System (NIPS) Conference*, September 1998.

[187] N. Vasconcelos and A. Lippman. Learning from user feedback in image retrieval systems. In *Advances in Neural Information Processing Systems*, volume 12, pages 977–986, Denver, Colo, USA, 1999.

[188] N. Vasconcelos and A. Lippman. A probabilistic architecture for content-based image retrieval. *Proc. IEEE CVPR*, 2000.

[189] N. Vasconcelos and A. Lippman. Statistical models of video structure for content analysis and characterization. *IEEE Trans. on IP*, 9(1):3–19, 2000.

[190] J.J. Verbeek, J.R.J. Nunnink, and N. Vlassis. Accelerated em-based clustering of large datasets. *Data Mining and Knowledge Discovery*, 13(3):291–307, 2006.

[191] J.J. Verbeek, N. Vlassis, and B. Krose. Efficient greedy learning of gaussian mixture models. *Neural Computation*, 15(2):469–485, 2003.

[192] J. Vermaak, P. Perez, and M. Gangnet. Rapid summarization and browsing of video sequences. In *British Machine Vision Conference*, Cardiff, September 2002.

[193] N. Vlassis and A. Likas. A greedy em algorithm for gaussian mixture learning. *Neural Processing Letters*, 15(1):77–87, 2002.

[194] N. Vlassis, Y. Sfakianakis, and W. Kowalzcyk. Gossip-based greedy gaussian mixture learning. *Proceedings 10th Panhellenic Conference on Informatics*, 3746:349–359, 2005.

[195] C. Wu. On the convergence properties of the em algorithm. *Annals of Statistics*, 11:95–103, 1983.

[196] L. Xie and P. Perez. Slightly supervised learning of part-based appearance models. In *Proc. of IEEE Workshop of learning in computer vision and pattern recognition*, Washington, DC, USA., June 2004.

[197] Y. Yacoob and J. Black. Parametrized modeling and recognition of activities. In *Sixth IEEE Int. Conf. on Computer Vision*, pages 120–127, Bombay, India, 1998.

[198] C. Yang, M. Dong, , and F. Fotouhi. Semantic feedback for interactive image retrieval. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 415–418, Singapore, 2005.

[199] L. Zhang, L. Chen, M. Li, and H. Zhang. Automated annotation of human faces in family albums. In *Proc. ACM Multimedia*, Berkeley, USA, 2003.

Contents

Estimation de modèles de mélange probabilistes: une proposition pour un fonctionnement réparti et décentralisé

Afshin NIKSERESHT

Résumé

Cette thèse traite de l'estimation statistique distribué, avec la motivation de, et l'application à l'indexation multimédia par le contenu. Les algorithmes et les données de divers contributeurs coopéreront vers un apprentissage statistique collectif. La contribution est un arrangement pour estimer une densité de probabilité multivariable, dans le cas où cette densité prend la forme d'un modèle de mélange gaussien. Dans ce cadre, l'agrégation des modèles probabilistes de mélanges gaussiens de la même classe, mais estimés à plusieurs nœuds sur différents ensembles de données, est une nécessité typique à laquelle nous nous intéressons dans cette thèse. Les approches proposées pour la fusion de mélanges gaussiens exigent uniquement le calcul modéré à chaque nœud et peu de données de transit entre les nœuds. Ces deux propriétés sont obtenues en agrégeant des modèles via leurs (peu) paramètres plutôt que par les données multimédia. Dans la première approche, en supposant que les mélanges sont estimés indépendamment, nous propageons leurs paramètres de façon décentralisée (gossip), dans un réseau, et agrégeons les modèles à partir des nœuds reliés entre eux, pour améliorer l'estimation. Les modèles de mélange sont en fait concaténés puis réduits à un nombre approprié de composants gaussiens. Une modification de la divergence de Kullback conduit à un processus itératif pour estimer ce modèle agrégé. Afin d'apporter une amélioration, l'agrégation est réalisée par la modélisation bayésienne du problème de groupement de composant de modèle de mélange gaussien et est résolue en utilisant la méthode variationnelle, appliquée au niveau de composant. Cela permet de déterminer, par un processus simple, peu coûteux et précis, les attributions des composants qui devraient être agrégés et le nombre de composants dans le mélange après l'agrégation. Comme seulement les paramètres du modèle sont échangés sur le réseau, le calcul et la charge du réseau restent très modérés.

Mots-clés : apprentissage distribué, calcul réparti, estimation distribuée, modèles de mélanges gaussiens (GMM), indexation multimédia, méthode variationnelle de l'apprentissage bayésien

Abstract

This thesis deals with the distributed statistical estimation, with its motivation from, and application to, multimedia content-based indexing. Algorithms and data from various contributors would cooperate towards a collective statistical learning. The contribution is a scheme for estimating a multivariate probability density in the case where this density takes the form of a Gaussian mixture model (GMM). In this setting, aggregation of probabilistic Gaussian mixture models of the same class, but estimated on several nodes on different data sets, is a typical need, which we address in this thesis. The proposed approaches for fusion only requires moderate computation at each node and little data to transit between nodes. Both properties are obtained by aggregating models via their (few) parameters, rather than via multimedia data itself. In the first approach, assuming independently estimated mixtures, we propagate their parameters in a decentralized fashion (gossip) in a network, and aggregate GMMs from connected nodes, to improve estimation. Mixture models are in fact concatenated, then reduced to a suitable number of Gaussian components. A modification on Kullback divergence leads to an iterative scheme for estimating this aggregated model. As an improvement through a change of principle over the first work, aggregation is achieved through Bayesian modelling of the GMM component grouping problem and solved using a variational Bayes technique, applied at component level. This determines, through a single, low-cost yet accurate process, assignments of components that should be aggregated and the number of components in the mixture after aggregation. Because only model parameters are exchanged on the network, computational and network load remain very moderate.

Keywords: distributed learning, distributed computing, distributed estimation, Gaussian Mixture Model (GMM), multimedia indexing, variational Bayesian learning