N° d'ordre : 3824

UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

THESE

Présentée en vue de l'obtention du grade de

DOCTEUR

Spécialité

AUTOMATIQUE & INFORMATIQUE INDUSTRIELLE

par

Habiboulaye AMADOU BOUBACAR

CLASSIFICATION DYNAMIQUE DE DONNEES NON-STATIONNAIRES APPRENTISSAGE ET SUIVI DE CLASSES EVOLUTIVES

Soutenue le 28 Juin 2006

Jury d'examen

Rapporteurs	:	C. AMBROISE	HDR, Maître de Conférences Université de Technologies de Compiègne
		Y. LECHEVALLIER	Directeur de Recherche INRIA, Rocquencourt
Examinateurs	:	J. P. CASSAR	Professeur Ecole Polytechnique Universitaire de Lille
		M. DAVY	HDR, Chargé de Recherche CNRS Laboratoire Automatique Génie Informatique et Signal
		C. DJERABA	Professeur Laboratoire d'Informatique Fondamentale de Lille
Directeur de thèse	:	S. MAOUCHE	Professeur, Directeur UFR-EEA Université de Sciences et Technologies de Lille
Co-encadrant	:	S. LECOEUCHE	HDR, Enseignant-chercheur Ecole Nationale Supérieure des Mines de Douai

Laboratoire Automatique, Génie Informatique & Signal - UMR 8146 - CNRS

A mes parents

Remerciements

Je voudrais tout d'abord exprimer ma reconnaissance à mon directeur de thèse, le Professeur Salah MAOUCHE, pour ses précieux conseils, ses encouragements et pour tout le temps qu'il m'a consacré. J'exprime ma profonde gratitude à Monsieur Stéphane LECŒUCHE d'avoir orienté mes travaux de recherche après mon DEA et co-dirigé cette thèse. Je le remercie également d'avoir structuré et amélioré la qualité de ce travail.

Je remercie Messieurs Christophe AMBROISE et Yves LECHEVALLIER pour l'intérêt qu'ils portent à mon travail en acceptant d'être les rapporteurs de cette thèse. Je remercie également Messieurs Manuel DAVY, Jean-Philippe CASSAR et Chabane DJERABA de m'avoir fait l'honneur de participer au jury en tant qu'examinateurs.

J'adresse mes sincères remerciements à Monsieur Philippe HAZEBROUCQ, Chef du département GIP de l'Ecole de Mines de Douai et Monsieur Philippe DESODT, Responsable de l'équipe SID pour m'avoir accueilli dans leurs locaux et me confier des activités d'enseignement de façon régulière sur toutes les années de cette thèse. Je leur exprime toute ma reconnaissance pour le poste de vacataire qu'ils m'ont accordé à l'Ecole de Mines de Douai pendant ma dernière année de thèse.

De même, j'exprime un remerciement tout particulier à Monsieur Jacques BOONAERT pour l'aide qu'il m'a apporté dans la résolution de certains problèmes techniques et scientifiques rencontrés dans mes travaux de thèse. Merci à Christiane, Pierre, Rosa, Antoine, Philippe, Frédéric, Gabriel, Martine et François pour l'ambiance conviviale qu'ils ont su créer au sein du département.

Je remercie également tout le personnel du Laboratoire LAGIS de Lille 1 où j'ai effectué cette thèse. J'exprime toute ma gratitude à Salah, Afzal, Sharam, Haoping, Jacques, Olivier, Pierre, Annick et tout ce qui m'ont apporté leur soutien et leur aide dans la préparation de cette thèse.

Aujourd'hui, j'ai une pensée profonde au défunt, Monsieur Emmanuel LECLERCQ, ancien directeur de l'EIPC pour la confiance qu'il m'a accordée dès mon arrivée en France ; grâce à son aide et son soutien considérable, j'ai pu effectuer mon ingéniorat et poursuivre des études de doctorat. Je salue également Martine POTEY et tout le personnel de l'EIPC.

Je ne peux terminer mes remerciements sans exprimer ma sincère gratitude à mes parents, mes frères et sœurs pour leur affection et leurs encouragements.

Enfin, je remercie du fond de mon cœur, Moinaecha pour son soutien considérable et tous mes amis pour les meilleurs moments de loisir, de détente et de joie passés ensemble.

ABREVIATIONS ET NOTATIONS

CDL	:	Clustering & Labelling Detection
FMMC	:	Fuzzy Min-Max Classification/Clustering
ESOM	:	Evolving Self-Organizing Map
CEM	:	Classification Expectation-Maximisation
AUDyC	:	AUto-adaptive & Dynamical Clustering
SAKM	:	Self-Adaptive Kernels Machine
SVM	:	Support Vectors Machines
RKHS	:	Reproduising Kernel Hilbert Space
NORMA	:	Naïve On-line R_{reg} Minimisation Algorithm
RBF	:	Radial Basic Function
RdF	:	Reconnaissance de Formes
SAD	:	Système d'Aide à la Décision

i	:	Indice de l'ordre de traitement des données spatiales (non temporelles)
t	:	Indice de temps, instant d'acquisition des données temporelles
m	:	Numéro de classe
j	:	Numéro de prototype ou de vecteur support
D	:	Dimension des données
L	:	Nombre de données
M	:	Nombre de classes
J	:	Nombre de prototypes
τ	:	Nombre de vecteurs support
χ	:	Espace de données, espace d'entrées
Γ	:	Espace de Hilbert à noyau reproduisant ou RKHS
X_i	:	Donnée spatiale
X_t	:	Donnée spatio-temporelle ou donnée non-stationnaire
$\Re$	:	Espace des valeurs réelles
Ω	:	Espace de paramètres de modèles
Π	:	Partition, ensemble de classes
$\Im$	:	Modèle de classification (ou de connaissances), ensemble des modèles de classes
C	:	Classe
ψ	:	Modèle de classe
Θ	:	Paramètres de classe (ou de composante)
φ	:	Fonction de densité de probabilité
π	:	Probabilité *a priori*, proportion dans le mélange
θ	:	Elément du vecteur Θ, paramètre de prototype, paramètre de vecteur support
$\overline{X}$	:	Moyenne du prototype gaussien
Σ	:	Matrice de covariance du prototype gaussien

π	:	Probabilité *a priori* (proportion dans un mélange)
p	:	Possibilité (proportion dans un mélange)
f	:	Fonction d'apprentissage à noyau
SV	:	Vecteur support de classe
α	:	Poids de la donnée (coefficient de Lagrange) dans le modèle de classe
ρ	:	Offset de la fonction d'apprentissage
ϕ	:	Transformation de l'espace d'entrée χ vers l'espace Hilbert Γ
Π^{win}	:	Ensemble de classes gagnantes
χ^{amb}	:	Ensemble de données ambiguës (partagées par 2 ou plusieurs classes)
χ^{rej}	:	Ensemble de données rejetées
Z	:	Critère à minimiser ou à maximiser dans le cas général
$S(\bullet,\bullet)$	:	Mesure de similarité
$\mu(\bullet)$	:	Fonction d'appartenance
$L[\bullet]$	:	Critère de vraisemblance
$L_{poss}[\bullet]$	:	Fonction objectif de AUDyC : Vraisemblance possibiliste
$R[\bullet,\bullet]$	:	Fonction objectif de SAKM : Risque d'apprentissage
$\kappa(\bullet,\bullet)$	:	Fonction noyau
$\xi(\bullet,\bullet)$	:	Fonction coût : erreur d'apprentissage
$\hat{E}(\bullet\|P)$	:	Estimateur de densité sachant la loi de probabilité P

Table des matières

Introduction générale

Contexte et motivations

Le sujet étudié entre dans le contexte général de l'Apprentissage Automatique (*Machine Learning*). L'Apprentissage Automatique est le domaine de recherche scientifique qui tente de comprendre et de reproduire des facultés d'apprentissage à un système artificiel. Il s'agit plus précisément, de concevoir des systèmes autonomes capables d'extraire à partir des informations disponibles, les connaissances utiles servant à la prise de décision. L'Apprentissage Automatique est en effet, une branche de l'Intelligence Artificielle *(A.I.)*. Il s'est formalisé avec le rapprochement des disciplines du Traitement de Signal et de la Reconnaissance de Formes *(RdF)*. Ces disciplines ont des concepts théoriques élaborés à partir des méthodes statistiques et des modèles mathématiques inspirés des réseaux neurobiologiques. Grâce aux potentialités de ces techniques, les capacités des systèmes d'apprentissage ne se limitent plus à la simple tâche de mémorisation de connaissances. Ils disposent de la faculté de généralisation leur permettant d'apprendre de façon autonome de nouvelles connaissances (non connues *a priori*). Divers algorithmes d'apprentissage ont ainsi été développés pour la mise au point de Systèmes d'Aide à la Décision *(S.A.D)*. Ceux-ci sont de nos jours très utilisés dans de nombreuses applications : supervision des processus, reconnaissance vocale, reconnaissance de visages, ...etc.

Lorsque les connaissances (déjà apprises) évoluent dans le temps, il est important que le système d'apprentissage automatique soit capable de s'adapter en prenant en compte ces évolutions. Très peu de systèmes d'apprentissage sont dotés de cette faculté. Kasabov [2003] introduit l'expression d'Intelligence Évolutionniste *(E.I.)* pour désigner une nouvelle branche de l' *A.I.* dédiée aux systèmes d'apprentissage capables de mémoriser, de généraliser et de suivre les évolutions. Ces systèmes, souvent dits d'apprentissage continu, adaptatif, ou itératif sont très utiles pour la résolution de nombreux problèmes. En effet, la plupart des processus naturels ou artificiels ont des comportements évolutifs.

Cadre de la thèse

Cette thèse s'inscrit dans le cadre de la Classification Dynamique de données non-stationnaires et se positionne comme le prolongement des travaux de thèse initiés par [Lurette, 2003]. La classification dynamique est l'axe de l'apprentissage automatique qui s'intéresse aux problèmes de classification et de modélisation adaptative des données non-stationnaires. Ces données caractérisant l'état des processus évolutifs sont également appelées par certains auteurs des données évolutives. En classification dynamique, les connaissances utiles à extraire (à apprendre) des données sont représentées par des classes susceptibles d'évoluer dans le temps. Afin d'introduire et de mieux appréhender cette problématique, on considère le processus représenté sur la figure 1. On souhaite extraire et modéliser de façon adaptative les

connaissances utiles caractérisant ce processus afin de suivre ses évolutions. Il faut d'abord réaliser la modélisation de l'état du processus à chaque instant (figure 1.a). Pour atteindre cet objectif, deux approches existent. La première se base sur l'extraction d'un modèle physique identifié en utilisant les entrées et sorties du processus [Ljung, 1999]. Cette démarche est intéressante lorsque le modèle physique existe et que les incertitudes de modélisation n'altèrent pas sa fidélité à l'état du processus. Dans la seconde, on extrait les informations pertinentes sensibles à l'état du processus en utilisant des signaux de capteurs [Mobley, 1992]. A partir de ces signaux et grâce à l'utilisation d'outils de traitement de signal, on construit un vecteur forme caractérisant l'état du processus. Cette démarche convient particulièrement aux processus complexes (difficiles à identifier), mais présente un risque important de fournir un vecteur forme redondant ou non pertinent. La mise en place d'un vecteur forme fidèle à l'état du processus et moins sensible aux perturbations est en général rendue possible grâce à des techniques de sélection, d'extraction et de réduction de dimension [Thiria & al., 1996].

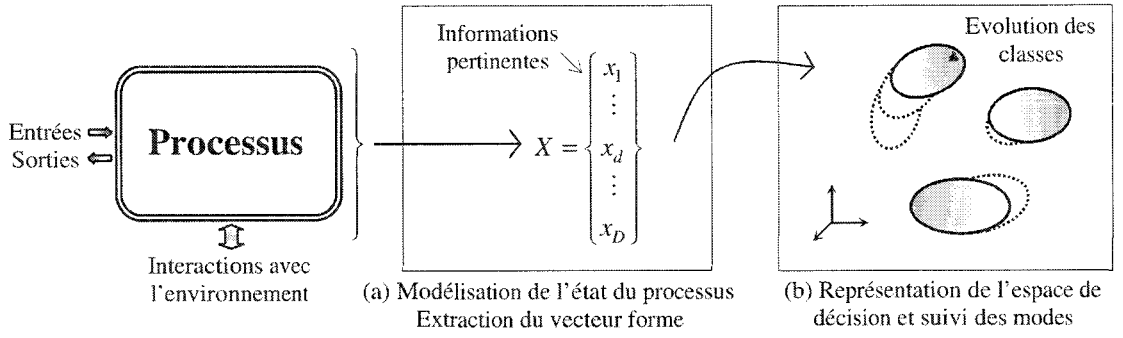

Figure 1 : Modélisation de l'état du processus et évolution de ses modes de fonctionnement

Maintenant, après la modélisation de l'état du processus, les connaissances apprises à partir des données, sont représentées par des classes. Celles-ci peuvent évoluer dans le temps en fonction du comportement du processus (figure 1.b). Par exemple, dans un contexte de supervision de processus industriels, ces classes correspondent aux modes de fonctionnement [Mobley, 1992; Boudaoud, 1998; Mouchaweh, 2002]. Les défaillances (pannes franches ou dégradations lentes) sur de tels processus se manifestent dans l'espace de décision par des changements brusques ou des évolutions d'un ou plusieurs modes de fonctionnement. Cet exemple montre l'intérêt de disposer d'outils capables d'apprendre (mémoriser et généraliser) les connaissances utiles extraites du processus et de suivre leurs évolutions dans le temps.

En utilisant les techniques avancées de traitement de signal, les principaux objectifs recherchés dans la mise en œuvre de ces algorithmes de classification dynamique sont les suivants :

- Apprentissage en ligne (*on-line learning*) : les informations sont traitées de façon séquentielle (incrémentale). Au fur et à mesure que les données se présentent, elles sont incorporées dans le modèle de connaissances et contribuent à sa redéfinition. Ceci doit s'effectuer sans réapprentissage du modèle avec les données déjà utilisées.

- Apprentissage non supervisé (*unsupervised learning*) : L'apprentissage se fait de manière complètement autonome sans l'aide d'un superviseur. Le système d'apprentissage doit être capable de déterminer, uniquement à partir des données, toutes les connaissances utiles (étiquettes de données, modèles de classes).
- Apprentissage adaptatif (*adaptive learning*) : L'adaptation récursive du modèle de connaissances doit se faire de façon à prendre en compte toutes les modifications et évolutions du modèle de connaissances. Ceci nécessite la modélisation continue de données non-stationnaires et le suivi de classes évolutives. D'autres phénomènes plus complexes entraînent la fusion, la scission ou la remise en cause des connaissances, etc.

Les objectifs énoncés ci-dessus montrent de façon succincte la complexité qui réside dans la mise en œuvre d'un système d'apprentissage continu et adaptatif. La problématique de la classification dynamique est rarement étudiée dans la littérature. Elle est abordée en détail dans ce manuscrit.

Contribution de la thèse

Les travaux effectués dans cette thèse sont largement consacrés à la recherche et à la mise au point d'algorithmes de classification dynamique. Ces algorithmes apportent des solutions intéressantes pour résoudre un certain nombre de problèmes de classification dynamique et de suivi d'évolutions en environnement de données non-stationnaires. Ils peuvent fonctionner hors ligne ou en ligne selon que les données d'apprentissage sont préalablement disponibles ou non. Les contributions essentielles de nos travaux de thèse sont :

- Description générique de classifieurs dynamiques : Une description générique a été introduite pour la conception d'algorithmes de classification dynamique. Afin de lui conférer des règles d'apprentissage et d'auto-adaptation, cette description générique est décrite avec une architecture neuronale évolutive permettant la création, l'adaptation, et l'élimination de classes. Deux algorithmes sont conçus à partir de cette description générique.
- Nouvelle version de l'AUDyC (*AUto-Adaptive et Dynamical Clustering*) : Cette nouvelle version de l'AUDyC utilise un modèle de classification inspiré du modèle de mélange gaussien. Il est décrit suivant une approche multimodale afin d'accroître les capacités de l'algorithme pour la modélisation des classes complexes. Le processus d'apprentissage de l'AUDyC est construit en quatre procédures : création, adaptation, fusion, évaluation (scission et élimination).
- Algorithme SAKM (*Self-Adaptive Kernel Machine*) : Cet algorithme est une contribution originale pour les SVM et méthodes à noyau dans les problèmes d'apprentissage en environnement de données non-stationnaires. Les classes du SAKM sont modélisées à l'aide de fonctions d'apprentissage à noyau. Ce sont des modèles non-paramétriques capables de représenter des formes complexes. Comme celui de l'AUDyC, le processus d'apprentissage du SAKM est construit en quatre procédures, mais ne comporte pas de mécanisme de scission.

Grâce à leurs règles d'auto-adaptation en environnement de données non-stationnaires, les algorithmes AUDyC et SAKM sont capables de prendre en compte les modifications du modèle de classification et d'effectuer le suivi de classes évolutives.

- Conception de Systèmes d'Aide à la Décision *(SAD)* : A partir des algorithmes de classification dynamique développés, on peut concevoir divers SAD. C'est ainsi qu'en couplant ces algorithmes avec des outils de détection, nous avons proposé un système de surveillance d'un processus industriel. De même, en utilisant des techniques d'identification récursive en amont de l'AUDyC, un système de suivi de processus non-stationnaires est proposé pour la détermination en ligne de son mode de fonctionnement.

Organisation du mémoire

Ce mémoire de thèse est structuré en cinq chapitres résumés comme suit :

Dans le chapitre 1, on présente les aspects de la classification dynamique. Trois parties constituent ce chapitre. La première partie détaille la problématique de la classification dynamique. La seconde partie expose quelques algorithmes ayant des propriétés intéressantes pour la classification de données non-stationnaires. Dans la dernière partie, nous développons une description générique pour la conception de classifieurs dynamiques.

Dans le chapitre 2, on présente les résultats d'une étude bibliographique de deux méthodes statistiques qui ont des bonnes performances en classification. Les modèles de mélange sont étudiés dans la première partie de ce chapitre. Doté d'un formalisme théorique très élaboré, les modèles de mélange sont des outils puissants souvent utilisés en classification de données spatiales. La seconde partie du chapitre est consacrée aux SVM et méthodes à noyau, plus particulièrement à l'étude de l'estimateur de densité à l'aide des méthodes à noyau. Ces méthodes sont parmi les plus récentes des méthodes statistiques. Elles ont des bons fondements théoriques et offrent des résultats expérimentaux très attractifs.

Malgré leurs grandes capacités de modélisation, très peu de travaux sont réalisés pour permettre l'utilisation des modèles de mélange ou des fonctions d'apprentissage à noyau en environnement de données non-stationnaires. Le chapitre 3 présente quelques techniques de modélisation adaptative. Ce chapitre apporte les outils nécessaires pour permettre la mise en œuvre des règles d'apprentissage et d'adaptation pour le classifieur dynamique. Dans un premier temps, les méthodes récursives basées sur le gradient stochastique sont présentées dans ce chapitre. Ensuite, on introduit les méthodes de mise à jour exacte. Enfin, quelques techniques de fusion et de scission de classes sont exposées.

Dans le chapitre 4, deux algorithmes sont développés suivant la description générique proposée au chapitre 1. Le premier algorithme est une nouvelle version de l'AUDyC qui utilise un modèle de mélange inspiré du mélange gaussien multimodal. Le second algorithme, le SAKM, se base quant à lui sur les méthodes à noyau. Après la présentation de leurs modèles de classification, les procédures d'apprentissage de ces deux algorithmes sont détaillées dans ce chapitre. De même, on y étudie leurs performances. Une synthèse théorique

est donnée à la fin de ce chapitre dans le but de comparer les performances des deux algorithmes.

Le chapitre 5 est consacré aux résultats expérimentaux. Ce chapitre est divisé en 5 parties. La première présente des tests sur le réglage des paramètres des algorithmes. La deuxième expérimente les procédures d'apprentissage des deux classifieurs dans le but d'illustrer leurs capacités d'auto-adaptation. Dans la troisième partie de ce chapitre, on analyse la complexité algorithmique d'un point de vue pratique. Une synthèse pratique est enfin élaborée pour comparer les deux algorithmes. Pour terminer ce chapitre, une dernière partie propose deux Systèmes d'Aide à la Décision (SAD) conçus à partir de nos deux classifieurs dynamiques. Il s'agit notamment d'un système de surveillance de processus industriels et d'un système de suivi en ligne de processus non-stationnaires.

On conclut ce mémoire en récapitulant les points essentiels du travail effectué et en donnant quelques perspectives de recherche et d'applications.

Chap. 1 : Aspects de la Classification Dynamique

1.1 Introduction

Dans le domaine de la classification, l'utilisation de la plupart des algorithmes développés jusqu'à présent nécessite que les données soient complètement disponibles dans un ensemble de cardinalité finie. Certains de ces algorithmes ont pour objectif de construire des classes en connaissant *a priori* les étiquettes d'appartenance des données aux classes : On parle de **classification supervisée**. Si les étiquettes des données ne sont pas connues *a priori*, l'algorithme a la double tâche de déterminer l'appartenance des données aux classes et construire la partition en utilisant généralement un critère de ressemblance : Il s'agit alors de **classification non-supervisée**. Selon les objectifs, la partition recherchée peut être déterminée suivant deux approches : L'**approche par modélisation** visant à estimer un modèle fidèle de classes (fonction contour ou de densité) et l'**approche de séparation** ayant pour but d'optimiser les frontières de décision séparant les classes [Fugunaga, 1990, Milgram & al., 2004 ; Halgamuge & Wang, 2005]. La première approche représente les classes sous forme d'agrégations de données dans des domaines fermés, la seconde quant à elle, n'exclut pas leur représentation dans des domaines ouverts (régions infinies).

Considérons maintenant les deux situations suivantes. Une première situation où l'ensemble des données à traiter est de taille trop importante et une seconde situation dans laquelle la base d'apprentissage est incomplète. Dans ces deux situations, les algorithmes de classification classiques présentent des limitations. Dans la première situation, les calculs deviendront trop importants voire irréalisables, à cause de la complexité algorithmique. Dans cette situation, des solutions basées sur des méthodes sélectives existent et visent à utiliser uniquement les données les plus pertinentes pour effectuer l'apprentissage [Tong , 2001]. La difficulté réside évidemment dans la mise en œuvre du critère de sélection qui risque d'écarter certaines informations importantes. Dans la seconde situation, les données sont acquises en ligne, de façon séquentielle. Toutes les données ne sont donc pas connues *a priori*, elles deviennent disponibles au fur et à mesure pour enrichir cette base. C'est le cas dans les applications en ligne où l'objectif consiste à déterminer le modèle de classification avec les données disponibles à chaque instant, sans attendre que la base d'apprentissage soit complétée

[Jin & al., 1999 ; Mouchaweh, 2002]. Ceci amène à redéfinir le modèle de classification à chaque fois qu'une nouvelle donnée se présente : C'est la **mise à jour** de modèle. Deux stratégies existent pour effectuer cette mise à jour, qui consiste en l'intégration des nouvelles informations pour adapter la partition. La première solution très basique propose de recalculer complètement le modèle de classification à chaque acquisition à partir de l'ensemble des données déjà connues auquel on ajoute la nouvelle donnée. Cette technique, connue en anglais sous le nom de *batch learning*, se trouve être très gourmande en temps de calcul et ne tire aucun avantage des connaissances extraites au cours des apprentissages précédents. La seconde solution consiste à utiliser les techniques d'**apprentissage incrémental** développées avec des **règles de mise à jour récursives**. Grâce à ces techniques, les informations portées par les nouvelles données sont incorporées séquentiellement dans le modèle de classification sans réutiliser (ou très peu) les anciennes données ; l'estimation se fait de façon récursive en utilisant les nouvelles informations et les connaissances déjà extraites sous forme de modèles de classes. En classification, les techniques d'apprentissage incrémental sont une alternative aux méthodes classiques. Elles disposent d'aptitudes caractéristiques pour faire face aux difficultés rencontrées dans les applications en ligne et dans les problèmes hors ligne où le nombre de données à traiter est trop important.

Dans la littérature, il existe d'innombrables méthodes de classification supervisée, non-supervisée et quelques méthodes combinant les deux stratégies [Liu & al., 2000]. De même, de nos jours, de plus en plus de travaux portent beaucoup d'intérêt aux techniques de classification incrémentale [Polikar, 2001 ; Mouchaweh & al., 2002]. Cependant, la plupart de ces algorithmes de classification incrémentale ont pour rôle essentiel de construire et d'adapter localement les modèles de classification sans pour autant prévoir leurs évolutions dans le temps, ou encore des situations de fusion de connaissances ou de remise en cause d'informations... Or, de tels phénomènes sont courants lorsqu'il s'agit de classification de **données non-stationnaires**. Certains auteurs emploient le terme de données évolutives [Loonis & Locteau, 2004]. Nous introduisons alors la définition suivante :

> **Définition 1.1:** les données non-stationnaires sont des données issues d'un processus dont le comportement est susceptible d'évoluer dans le temps. La non-stationnarité entraîne une évolution des caractéristiques des modèles de connaissances (classes) apprises de ces données. En classification, nous parlons de classes évolutives.

Aujourd'hui, beaucoup d'intérêts sont portés sur la manipulation de données non-stationnaires. En classification comme dans divers autres domaines, ces données restent très difficiles à traiter à cause de l'évolution du modèle de connaissances. L'expression de **classification dynamique** est ainsi introduite pour qualifier les algorithmes développés pour la classification automatique de ces données [Lurette, 2003, Lecoeuche, 2006].

Dans la première partie de ce chapitre, nous allons étudier la problématique de classification dynamique. Cette partie a pour but de montrer les difficultés liées à la classification de données non-stationnaires et les différentes situations d'évolution engendrées par de telles données. Elle se termine par un exposé sur les performances que doit avoir un

classifieur dynamique pour accomplir sa tâche. La seconde partie est consacrée à une étude bibliographique dans laquelle nous nous intéressons plus particulièrement aux algorithmes ayant des propriétés intéressantes pour la classification dynamique. Nous présentons leurs principes de mise en œuvre avant de montrer leurs limitations. Enfin, dans la dernière partie nous proposons une description générique d'algorithmes de classification dynamique. Cette description est une contribution importante de cette thèse. Elle introduit le formalisme général sur lequel est fondé notre démarche de conception de classifieur dynamique.

1.2 Problématique de la classification dynamique

La problématique de classification dynamique regroupe les problèmes de classification non-supervisée, d'apprentissage séquentiel (en ligne) et de modélisation adaptative. A toutes ces difficultés, il faut ajouter celles liées aux phénomènes dynamiques dans un environnement de données non-stationnaires. Il s'agit, entre autres, de mécanismes d'apparition et de de disparition de classes, d'évolution de classes, de fusion et de scission des classes. Cet exposé sera plus largement consacré à ces phénomènes dynamiques qui se révèlent comme une problématique relativement nouvelle et rarement abordée. Mais tout d'abord, nous commençons par présenter brièvement les problèmes de classification non-supervisée, d'apprentissage séquentiel et de modélisation adaptative qui ont fait déjà l'objet d'études dans un certain nombre de travaux de classification.

Classification non-supervisée : ni les étiquettes (ou classes d'appartenance) des données, ni le nombre de classes ne sont connues *a priori*. La méthode de classification a donc pour objectif de construire la partition des données et de déterminer l'appartenance des données aux classes. L'affectation des données aux classes se fait généralement à partir de l'évaluation d'un critère de ressemblance. La définition d'un critère de ressemblance entre les données dépend de la nature du problème à traiter. Dans le cadre de la classification des données spatiales, le critère de ressemblance vise à affecter les données géographiquement proches (ou similaires) à la même classe. Cette mesure de similarité définie la structure des classes suivant les groupements naturels des données. Par exemple, l'utilisation d'une distance euclidienne conduit à des classes de modèle hyper-sphériques. Il faut cependant noter qu'en classification non-supervisée, les classes formées n'ont pas une interprétation significative vis-à-vis du problème à traiter.

Apprentissage séquentiel : Contrairement à l'apprentissage en base complète où toutes les données sont disponibles et accessibles à tout instant, dans un contexte d'apprentissage séquentiel, les données enrichissent progressivement la base d'apprentissage. Dans ces situations d'apprentissage à partir d'une base incomplète, la tâche de classification consiste à incorporer chaque donnée dès qu'elle se présente, afin d'actualiser le modèle de connaissances (classes). Cette tâche doit se faire en utilisant les connaissances extraites des données déjà acquises et sans attendre que toutes les informations contenues dans les données futures soient disponibles. C'est l'enjeu des techniques d'apprentissage incrémental.

Modélisation adaptative : Dans l'espace de classification, la partition des données est caractérisée par des classes ou regroupements de données similaires. Cette partition est généralement représentée par un modèle de classification. Le choix d'un modèle de classification adéquat pour modéliser la distribution des données est une difficulté qui sera étudiée dans la partie 1.4 de ce chapitre et dans le chapitre 2. Dans un contexte d'apprentissage incrémental, une autre difficulté se pose. Il s'agit de la mise à jour du modèle de classification au fur et à mesure que les données se présentent. Cette tâche requiert la mise en place de règles d'adaptation tenant compte des changements engendrés sur la partition au cours de l'incorporation des nouvelles données dans les classes. En effet, le modèle de classification subit des changements de nature plus ou moins complexe. Ceux-ci se traduisent par des modifications locales et/ou des évolutions du modèle de classification.

La plupart des techniques de modélisation adaptative proposées dans la littérature se limitent à la prise en compte des modifications locales de classes. Cependant, la problématique d'évolution des classes entraînant un changement de leurs régions de localisation, est rarement abordée. Il est donc nécessaire de disposer de règles d'apprentissage continu et de suivi d'évolution. Cette problématique est d'autant plus importante pour la classification dynamique que ses applications très variées concernent la plupart des processus ayant des comportements évolutifs. Nous y reviendrons dans la section suivante au cours de l'exposé des phénomènes dynamiques plus précisément dans la sous-section 1.2.1.2.

1.2.1 Phénomènes dynamiques en environnement non stationnaire

Cette section détaille les mécanismes généraux engendrés par l'incorporation des nouvelles données dans la partition et les phénomènes dynamiques causés par la non-stationnarité des données. Avant d'exposer ces mécanismes, nous allons tout d'abord donner les définitions d'une classe et d'une partition dynamique dans le contexte de notre étude.

> **Définition 1.2 :** Une classe est définie comme un ensemble (ou regroupement homogène) de données similaires. La similarité est évaluée d'un point de vue de la proximité géométrique des données. Lorsque les données d'une classe évoluent dans le temps, cette classe est dite évolutive. Son modèle doit donc être adapté.

De cette définition, on déduit que deux classes différentes ne se chevauchent pas ou très peu, sinon, leurs données seront suffisamment proches pour constituer une seule et unique classe.

> **Définition 1.3 :** Une partition dynamique représente l'ensemble des regroupements homogènes des données non-stationnaires. Elle est associée à un modèle de connaissances (i.e. classes) susceptible d'évoluer dans le temps en fonction de l'incorporation de nouvelles données.

Dans le cadre de l'étude de la problématique des données non-stationnaires, un exposé des divers mécanismes est présenté. Ces mécanismes concernent l'apparition de nouvelles classes, l'évolution de la partition dynamique, la fusion de classes et l'évaluation de la

représentativité du modèle de classification à travers la scission de classes et l'élimination de classes parasites ou obsolètes. Pour chacune des situations évoquées :

- On explique comment interagissent les nouvelles données avec la partition dynamique.
- On décrit les mécanismes engendrés par l'incorporation des données dans la partition.
- On montre les risques encourus par la méthode de classification dynamique.
- Et, on donne des exemples d'illustration.

1.2.1.1 Apparition de nouveautés dans un espace de classification

Lorsque de nouvelles données sont acquises dans une même région ou différentes régions non occupées par les classes de la partition existante et si ces données apportent suffisamment d'informations, elles révèlent l'apparition de nouvelles classes. L'apparition de chaque nouvelle classe se caractérise par un regroupement de données similaires dans une même région de l'espace.

Dans de telles situations, l'objectif consiste à la détection et à la **création des classes** constituées par ces nouvelles données. Ceci se traduit par la double tâche de déterminer, d'une part, les données suffisamment proches pour former chaque classe et d'autre part, de calculer les modèles des nouvelles classes créées. La première tâche est généralement accomplie par l'utilisation d'un critère de similarité, tandis que la seconde nécessite la mise en œuvre d'une procédure d'initialisation respectant la distribution des données. Au cours de ce mécanisme de création, la partition dynamique est actualisée par l'insertion des nouvelles classes.

Nous recensons trois cas de figure pour caractériser l'apparition de nouveautés :

Cas 1) Apparition de classes dans un espace de classification « vide »

Au départ de l'apprentissage, les premières données arrivent dans l'espace de classification. Or, cet espace étant initialement « vide », aucune classe n'existe encore. Si les premières données acquises sont similaires (géographiquement proches), elles constituent une classe, sinon elles révèlent l'apparition de plusieurs classes disjointes compte tenu de la définition 1.2. La figure 1.1.a présente une situation de création d'une classe dans un espace de classification initialement « vide ».

Cas 2) Apparition de nouvelles classes après un changement brusque

Dans cette situation, une ou plusieurs classes sont déjà créées dans l'espace de classification. Des nouvelles données apparaissent soudainement dans une région isolée de la partition constituée par les classes existantes. Ces nouvelles données, si elles sont similaires et suffisantes, révèlent donc l'apparition d'une nouvelle classe (figure 1.1.b).

Cas 3) Apparition de nouvelles classes après une évolution rapide de données

Dans certaines situations, l'apparition d'une classe peut être précédée par une évolution plus ou moins rapide de données. Ces données de transition sont considérées comme des

points caractérisant une dérive rapide entre deux classes. Dans le cadre de la classification, ces données de transition ne présentent pas une structure cohérente et stable pour contribuer à la formation d'une classe. Après la phase transitoire, lorsque que suffisamment de données se stabilisent dans une région isolée de la partition existante, ces données seront, quant à elles, révélatrices de l'apparition d'une nouvelle classe (figure 1.1.c). Dans certaines applications, l'analyse de cette phase transitoire décrite par la dérive de données entre deux classes présente un intérêt considérable notamment pour établir un pronostique [Amadou, 2002].

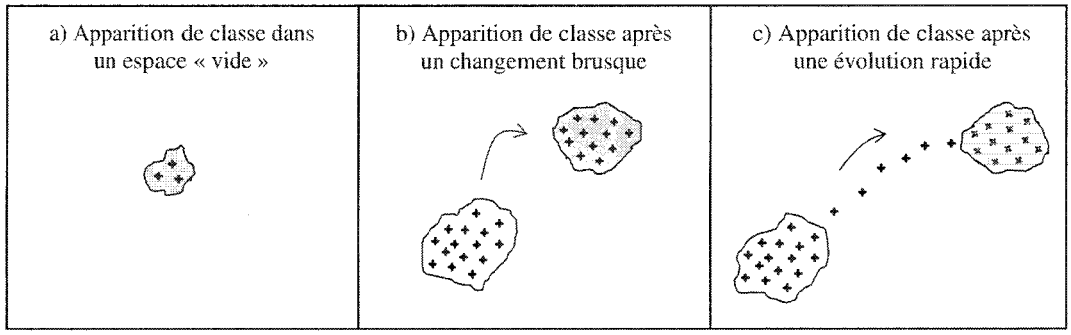

Figure 1.1 : (a) Apparition de classe dans un espace de classification « vide », (b) Apparition d'une nouvelle classe après un saut brut (c) Apparition d'une nouvelle classe après une dérive rapide.

Le phénomène d'apparition de nouvelles classes présente plusieurs risques. A cause de l'absence de superviseur, la méthode de classification peut affecter des données de classes différentes à une seule classe, ou encore attribuer des données d'une même classe à des classes différentes. Par ailleurs, la distribution des données n'étant pas connue *a priori*, l'initialisation d'un modèle de classe avec un nombre insuffisant de données reste difficile et peut conduire à des aberrations. D'autre part, à cause de l'influence du bruit et des perturbations, des erreurs de classification sont fréquentes. Ces erreurs peuvent conduire à la création de classes parasites affectant la qualité de modélisation. Néanmoins, ces classes parasites ne doivent pas être confondues avec des classes représentatives dont les données se présentent d'une façon périodique (par intermittence) dans une même région de l'espace.

1.2.1.2 Evolution de la partition dynamique

Contrairement au contexte de classification de données stationnaires (classes statiques), la problématique de la classification dynamique pose le problème d'évolution de la partition au cours de l'incorporation de nouvelles données. Cette problématique distingue notre étude de la plupart des travaux menés jusqu'à présent, qu'il s'agisse de classification non-supervisée ou incrémentale. Rappelons que l'intérêt de prendre en compte les évolutions des classes concerne diverses applications réelles : diagnostic industriel (évolution des modes de fonctionnement), médicale (expansion de cancers), surveillance vidéo (mouvement des cibles), ... etc. En effet, dans ce type d'applications, au fur et à mesure que les nouvelles données deviennent disponibles, leurs informations participent à la construction et à l'adaptation des classes. L'incorporation de ces données entraîne des évolutions plus ou moins complexes de la partition dynamique (figure 1.2.c).

Il est donc nécessaire dans ces situations de prendre en compte ces évolutions à travers l'adaptation des classes concernées. L'accomplissement de cette tâche passe par la réponse à deux questions : De quelle classe une nouvelle donnée est-elle suffisamment proche pour contribuer à la redéfinition de son modèle ? De quelle manière l'information portée par cette nouvelle donnée contribue-t-elle à adapter le modèle de la classe ? La réponse à la première question est généralement donnée par le critère de similarité qui détermine l'appartenance des données aux classes. Quant à la deuxième question, la réponse passe à travers la mise en œuvre de règle d'adaptation récursive.

Les situations d'évolution de la partition dynamique peuvent être divisées en deux cas.

Cas 1) Modifications locales de classes

Les modifications locales de classes sont des mécanismes dynamiques dus à l'incorporation de nouvelles données non-stationnaires. Elles se traduisent par des variations plus ou moins importantes observables localement au sein de la structure (géométrie et contour) de classes dans leur région de définition. Les grossissements, rétrécissements et rotations de classes sont les modifications les plus courantes (figures 1.2.a et 1.2.b). La plupart des méthodes de modélisation adaptative développées avec des règles d'apprentissage incrémental (sans oubli d'information), se limitent à la prise en compte des modifications locales. Au travers de ces règles, l'**adaptation de modèles de classes** se fait donc localement.

Cas 2) Evolutions avec glissements de classes

Nous parlons d'évolutions avec glissements de classe quand les nouvelles données d'une même classe s'éloignent progressivement de la région occupée par les anciennes données. Contrairement aux modifications locales, les évolutions avec glissements de classes s'accompagnent de changements de la région de définition des classes. Ces évolutions se font à travers de simples déplacements de classes ou la combinaison des déplacements avec des variations de la structure de ces classes. Dans ces situations, la classification des données non-stationnaires nécessite un **suivi des classes évolutives**. Ce suivi des classes évolutives passe nécessairement par l'intégration des informations récentes (représentatives) apportées par les nouvelles données dans la classe et l'élagage des informations obsolètes (non-représentatives) contenues dans les données les plus anciennes de cette même classe.

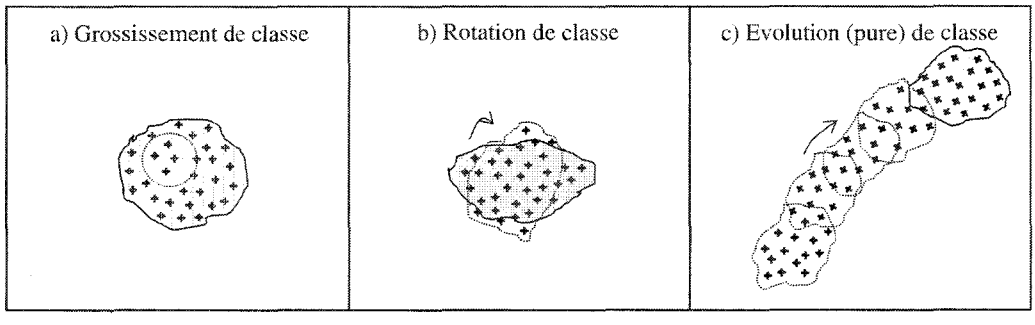

Figure 1.2 : (a) Développement de classe, (b) Modification quelconque de classe (c) Evolution de classe suite à une dérive de données non-stationnaires.

La mise à jour récursive des modèles de classes est rendue possible grâce à des règles de mise à jour récursives permettant à la fois **l'ajout et le retrait d'informations**. Le suivi des classes évolutives constitue l'un des objectifs essentiels de la classification en environnement dynamique. La figure 1.2.c illustre une classe évolutive caractérisée par une séquence d'états transitoires.

La mise en œuvre d'une méthode de mise à jour récursive est une épreuve quelque fois très difficile selon le modèle de classification utilisé. Dans les situations d'adaptation de classes, l'influence du bruit peut causer des imperfections sur la modélisation des classes. De même, lors de la modélisation dynamique, le phénomène de dérive de données (figure 1.1.c) ne doit pas être confondue avec une évolution de classe (figure 1.2.c). Par ailleurs, la dynamique des classes évolutives peut entraîner des interactions entre les classes (coalitions, séparations,...) pouvant affecter la qualité de la modélisation si celles-ci ne sont pas correctement traitées. C'est l'objectif des sous-sections 1.2.1.3 et 1.2.1.4.

1.2.1.3 Fusion d'informations mutuelles

Dans un environnement non-stationnaire, suite à certains phénomènes dynamiques, des interactions peuvent se produire entre deux ou plusieurs classes. Ainsi, une ou plusieurs classes initialement disjointes peuvent se rejoindre et partager des données. Par exemple, en reconnaissance de caractères, à cause de données manquantes, deux formes initialement disjointes peuvent fusionner pour constituer un caractère après l'acquisition de données complémentaires. Lorsque deux ou plusieurs classes différentes partagent des données, ces données sont porteuses d'informations mutuelles et sont dites **ambiguës** [Dubuisson, 1990]. Or, en se référant à la définition 1.2 présentant une classe comme un ensemble de données géographiquement proches, les classes partageant des données ambiguës correspondent à une même classe.

Afin de lever l'ambiguïté en cas d'informations mutuelles, la méthode de classification dynamique doit fusionner les classes concernées pour ne former qu'une seule classe (figure 1.3.a). La détection d'informations mutuelles peut être effectuée avec un critère de ressemblance. Par contre, la détermination du modèle résultant de la fusion, sans effectuer un réapprentissage sur toutes les données, nécessite la mise œuvre de **règles de fusion**.

La mise en œuvre d'une règle de fusion est une tâche souvent compliquée qui peut exposer la partition dynamique aux problèmes d'optima locaux comme le sous-apprentissage de modèle. Un autre risque est lié au bruit dans les données qui peut se comporter comme une fausse information mutuelle et déclencher une fusion de classes non souhaitée.

1.2.1.4 Séparation d'informations exclusives

Le phénomène inverse de la fusion peut se produire en environnement non-stationnaire lorsque une classe se scinde en deux classes au cours de l'incorporation des nouvelles données (figure 3.b). Par exemple, les données caractérisant le mode de fonctionnement d'un processus, peuvent être initialement mélangées avec des données parasites causées par des

perturbations extérieures. Lors de l'évolution de ce mode de fonctionnement, la classe de départ peut se scinder en deux suite à une séparation de la classe normale de la classe parasite.

Lorsque de tels phénomènes se produisent, la méthode de classification dynamique doit décider de la scission en utilisant un critère de détection de scission de classe et elle doit comporter des **règles de scission** de classe permettant la détermination des modèles des classes résultantes de la scission.

La difficulté principale concerne la mise en œuvre du critère de détection de scission. En effet, celui-ci doit répondre à la question : les données contenues dans chaque classe sont elles similaires ou non ? Un critère répondant de façon fiable et certaine à cette question reste évidement très difficile à mettre en œuvre. Or, si une classe est constituée par des données non similaires, il s'ensuit une mauvaise modélisation de la partition.

1.2.1.5 Suppression de connaissances obsolètes ou parasites

La classification dynamique s'effectue à travers la modélisation continue de processus sur de longues durées. Il se peut donc que les connaissances apprises dans un passé plus ou moins lointain ne soient plus valables à l'instant courant. Les classes représentant les connaissances obsolètes doivent donc être éliminées de l'espace de classification. De même, d'autres classes parasites créées à cause du bruit ou des perturbations extérieures, sont à supprimer (figure 1.3.c).

Un classifieur dynamique doit donc comporter une **procédure d'élimination** des classes non-représentatives (classes parasites ou obsolètes). La détection de classes parasites est généralement basée sur un *critère de cardinalité* [Lurette, 2003].

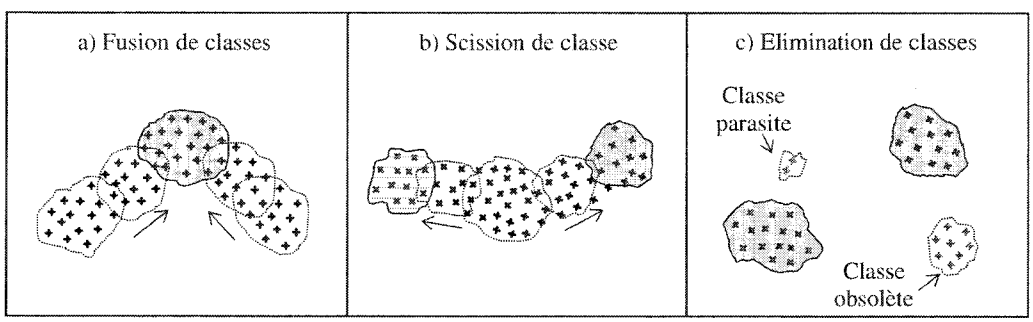

Figure 1.3 : (a) Fusion de classes partageant des informations mutuelles (b) Scission de classes après séparation des informations partagées (c) Elimination de classes obsolètes et parasites.

Cependant, la notion d'obsolescence de classes est très subjective et dépend fortement du contexte de l'application. De même, dans un espace de classification dynamique où la base d'apprentissage est incomplète, à quel moment peut on considérer qu'une classe est parasite ?

1.2.2 Performances de classifieurs dynamiques

Dans la section précédente, nous avons exposé les difficultés causées par la non-stationnarité des données et les phénomènes dynamiques engendrés dans le contexte de la

classification en environnement dynamique. Cependant, il ne suffit pas qu'un classifieur dispose de règles d'auto-adaptation en environnement dynamique pour résoudre tous les problèmes. La prise en compte de phénomènes dynamiques étudiés précédemment est une condition nécessaire qui doit se faire en respectant les contraintes du problème. Cela passe par la recherche de certaines performances. Dans la littérature, il existe un certain nombre de critères de performance pour les méthodes classiques de classification [Dongxin, 1999 ; McLachlan & Peel, 2002]. A l'inverse, la classification dynamique n'ayant été que récemment abordée, peu de critères spécifiques ont été établis. Par ailleurs, il est difficile de déterminer de façon exhaustive les critères de performance que doit satisfaire un classifieur dynamique. En effet, ces derniers dépendent fortement des contraintes et objectifs du problème. Néanmoins, nous allons tenter d'introduire quelques critères de performance qui nous semblent importants à considérer pour un classifieur dynamique.

1.2.2.1 Qualité de classification

En classification, les connaissances disponibles dans les données sont modélisées sous forme de classes en utilisant des modèles prédéfinis (exemple : modèles gaussiens) ou modèles induits par des mesures de similarité (exemple : distances euclidiennes). Rappelons que ces modèles peuvent être paramétriques pour des distributions supposées connues *a priori*, ou non-paramétriques lorsque ces distributions sont inconnues. Dans les deux situations, pour obtenir une meilleure qualité de modélisation, le classifieur dynamique doit satisfaire deux conditions :

> **Condition 1.1 :** La modélisation de classes doit se faire à travers le choix d'un bon modèle de classification. Le bon modèle est celui qui s'adapte parfaitement à la distribution des données dans chaque classe.

Cette première condition est nécessaire pour éviter les problèmes d'optima locaux. Par exemple, une classe de forme elliptique (modèle gaussien) ne convient pas à modéliser des données générées suivant une distribution bornée (hyper-cubique) car elle conduit à une situation de sur-apprentissage (*over-fitting*) ou de sous-apprentissage (*under-fitting*).

> **Condition 1.2 :** les erreurs de classification des données mesurées relativement à l'ensemble des classes, doivent être minimales. Ces erreurs sont définies en fonction du problème à traiter.

Cette deuxième condition ne peut être satisfaite que si la première est préalablement garantie ou satisfaite avec une certaine tolérance. En effet, si le modèle est mauvais, mal choisi, la minimisation d'erreur ne peut pas être garantie. La mesure d'erreur est plus facile lorsqu'on dispose de connaissances *a priori* sur les étiquettes réelles de données. Ce qui est loin d'être le cas dans un contexte de classification dynamique.

Pour mesurer la qualité de classification, certaines méthodes statistiques maximisent un *critère de vraisemblance* évaluée entre les données et le modèle de classification [Dempster & al., 1977] tandis que d'autres recherchent la minimisation d'un *risque* sur le classifieur

[Vapnik, 1998]. Dans le cadre de la classification dynamique, ces conditions seront recherchées tout au long de l'apprentissage afin de garantir la qualité de classification.

1.2.2.2 Convergence

La notion de convergence d'algorithme présente beaucoup d'intérêt dans le cadre de la classification dynamique utilisant des techniques d'apprentissage séquentiel (en ligne), continu et adaptatif. En effet, dans la plupart des applications, il s'agit de modéliser un processus sur de longues durées. L'étude de la convergence permet d'analyser le comportement du classifieur dynamique pendant un apprentissage en temps « infini ». Ceci se fait à partir de la détermination continue et de l'analyse de l'erreur instantanée dans un environnement non-stationnaire. Pour garantir de meilleures performances, il est important que les erreurs commises tout au long de l'apprentissage par l'outil de classification dynamique convergent vers une limite finie ou à défaut que celles-ci ne divergent pas. En effet, si les erreurs divergent, il est difficile de prévoir le comportement du système d'apprentissage, celui-ci peut devenir chaotique.

1.2.2.3 Complexité algorithmique

La complexité est une notion qui fait référence à l'ordre de grandeur du nombre d'opérations effectuées par un algorithme. La détermination de la complexité d'un algorithme consiste en l'estimation du coût précis de toutes les procédures de calcul. Cette grandeur a une importance considérable dans les applications en ligne et temps réel. En effet, la complexité permet d'évaluer la consommation du temps, la compacité (utilisation de la mémoire), et donc les ressources informatiques nécessaires au système d'apprentissage pour accomplir efficacement sa fonction. La détermination de la complexité devient une tâche très difficile pour des programmes longs et complexes. Par ailleurs, selon l'algorithme, la complexité peut varier en fonction de diverses grandeurs telles que : la quantité de données, leur dimension, ... etc. Dans le cadre de la classification dynamique, il est important de garantir que le temps de calcul pour traiter séquentiellement chaque nouvelle donnée soit inférieur au temps d'échantillonnage c'est-à-dire le temps entre deux acquisitions.

1.3 Classification dynamique et architectures évolutives

La conception d'algorithmes de classification dynamique est un art difficile compte tenu de la non-stationnarité de données et de l'évolution de classes. Dans l'histoire des méthodes de reconnaissance de formes, l'algorithme ISODATA est l'une des premières réalisations dotée de règles de fusion, de scission et d'élimination [Ball & Hall, 1965]. Dans la littérature plus récente, la majorité des travaux de recherche traitant des comportements évolutifs des processus s'inspirent des mécanismes d'évolution des formes de vies [Dessalles, 1996] ou des aspects neurobiologiques du cerveau humain [Arbib, 1987]. C'est ainsi qu'à travers l'étude de ces principes biologiques, des méthodes et modèles mathématiques ont été développés pour servir à la construction des systèmes informatiques pour des applications dans un

environnement dynamique (ou évolutionniste). Ces méthodes assez variées sont aujourd'hui utilisées en Apprentissage Artificielle, en Robotique, en Commande et Contrôle des processus, ...

Dans la section suivante, sont tout d'abord présentées des techniques basées sur les chaînes de Markov, puis les méthodes évolutionnistes (algorithmes génétiques) et les méthodes connexionnistes (réseaux de neurones). Ensuite, nous portons plus d'intérêts sur quelques algorithmes neuronaux à architectures évolutives qui sont introduits pour la classification incrémentale.

1.3.1 Généralités

Les chaînes de Markov sont des méthodes capables de modéliser des états d'évolution à travers une succession de transitions [Gordon, 1965 ; Kijima, 1997]. Un processus de Markov est constitué d'états, d'actions et de transitions pour représenter les conditions de changement et les séquences d'évolutions. Le passage entre les différents états est conditionné par une loi de probabilité. En se basant sur ces méthodes, Loonis et Locteau [2004] propose une approche markovienne pour la poursuite des classes évolutives. Cette approche se présente comme une technique de planification visant à choisir des agents de classification (association de classifieurs) adaptés pour modéliser les données non-stationnaires à chaque transition. L'apprentissage se fait étape par étape à travers le choix des agents de classification de telle sorte que lorsque les données évoluent, les modèles soient également soumis aux lois d'évolution. En utilisant les chaînes de Markov, le suivi de ces évolutions est obtenu en planifiant explicitement plusieurs modèles de classification ordonnés par un critère de sélection. Le déploiement d'agents de classification est basé sur le formalisme probabiliste d'un processus markovien défini sur un horizon fini ou infini dénombrable. Cet outil a été utilisé pour suivre les caractéristiques physiques d'une espèce de poissons suivant un cycle saisonnier dans le cadre d'un projet appelé eFISH [Loonis & Locteau, 2004]. Cette approche markovienne est robuste et conduit à une planification optimale des agents de classification. Cependant, elle est fondée sur un apprentissage hors ligne (base complète) dans un univers fermé. De même, la modélisation adaptative de la partition dynamique n'est pas faite de façon récursive, l'estimation des modèles de classes est effectuée par un algorithme de maximisation de vraisemblance au prix d'un coût de calcul très important.

Parmi les méthodes s'inspirant des aspects biologiques, les algorithmes génétiques sont introduits dès les années 60 par Holland [1975]. Le principe est basé sur l'exploitation de l'évolution naturelle des gènes. Dans l'espoir d'imiter l'évolution des organismes vivants, ces méthodes évolutionnistes tentent de reproduire l'évolution des gènes, la création de nouveaux individus et de nouvelles espèces à travers des mécanismes de croisements et des mutations [Dessalles,1996 ; Gorunescu & Dumitrescu, 2003]. A travers ces mécanismes, les algorithmes génétiques se trouvent dotés d'une véritable créativité pour la découverte des meilleures solutions. Cependant, il est nécessaire que le système garde en mémoire toutes les solutions potentielles qui seront ensuite tirées de façon aléatoire pour être croisées. Cette dernière

condition représente une limitation majeure dans un contexte de classification dynamique car il s'agit de problème d'apprentissage avec une base incomplète et sans aucune connaissance *a priori* des solutions potentielles dans l'espace de classification. En plus de ces contraintes incontournables dans un univers de discours ouvert, le mécanisme de croisement de toutes les solutions potentielles est souvent d'une grande complexité. Les algorithmes génétiques sont mieux adaptés aux problèmes d'optimisation où ils sont utilisés et ont prouvés leurs performances [Moon & al., 2002]. Néanmoins, il existe quelques travaux initiés pour la classification en environnement non-stationnaire. Dans [Gorunescu & Dumitrescu, 2003], une technique évolutionniste de classification automatique dotée des règles d'apprentissage incrémental et des procédures de fusion et de scission est présentée. La classification se base sur une fonction objectif qui est établie à partir d'un critère probabiliste pour mesurer la qualité de classification et évaluer l'apport d'information apportée par chaque donnée par rapport aux classes. Les procédures de fusion et de scission de cet algorithme sont basées sur des mécanismes de croisement et de mutation. Le résultat final de cet algorithme est indépendant de l'ordre d'arrivée des données contrairement à la plupart des méthodes de classification incrémentale. Cependant, cet algorithme n'a pas des capacités d'auto-adaptation permettant le suivi de classes évolutives. De même, l'approche probabiliste n'est pas adaptée à l'apprentissage à partir d'une base incomplète où l'univers de discours est ouvert.

Introduits par les travaux [Hebb, 1949 ; Rosenblatt, 1958], les réseaux de neurones sont aujourd'hui très utilisés dans les problèmes d'apprentissage. Ce sont des modèles connexionnistes basés sur une modélisation mathématique du neurone formel. Outils puissants d'analyse et d'interprétation de données, les réseaux de neurones sont dotés de grandes capacités d'adaptation. On distingue deux grands groupes suivant la structure du réseau : Les réseaux à **architectures statiques** qui sont généralement conçus pour un apprentissage supervisé. Le nombre de neurones de chaque couche est fixé *a priori* et ce nombre reste inchangé tout au long de l'apprentissage. L'apprentissage de ces réseaux est effectué avec des règles de rétro-propagation. Dans ce groupe, on retrouve la plupart des réseaux PCM (Perceptron Multi-Couche) [Jodouin, 1994 ; Osorio, 1998]. Le second groupe concerne les réseaux à **architectures évolutives**. Ces réseaux sont généralement dédiés à un apprentissage non-supervisé, mais ils peuvent aussi fonctionner en apprentissage supervisé. Grâce à la structure dynamique de ces réseaux, leur nombre de neurones et de connexions peut varier. Dans ce groupe, on a généralement des réseaux à prototypes qui servent à représenter des prototypes comme des unités regroupant les données de caractéristiques typiques [Fiesler, 1994]. L'évolution de leur architecture peut être de type *constructif* (incrémental) ou *destructif* (élagage). Le choix entre les deux stratégies est contreversé. Mais, d'un point de vue calculatoire, commencer petit et ajouter des neurones et des connexions est plus performant que de faire l'apprentissage en partant d'un grand réseau pour ensuite élaguer les éléments inutiles. De nouvelles techniques tendent à combiner les deux stratégies. Nous portons plus d'intérêt à ces réseaux à architectures évolutives pour leurs capacités d'auto-adaptation en environnement dynamique. Trois algorithmes basés sur des réseaux à structure dynamique sont présentés dans les sections 1.3.1, 1.3.2 et 1.3.3.

Avant de terminer cette étude de bibliographie générale, citons brièvement les travaux de thèse de Fabiani [1994, 1996] concernant les stratégies de traitement de l'information dans un environnement dynamique. Dans cette thèse, une étude théorique est proposée par l'auteur avec un formalisme très général sur l'évolution des croyances et les règles de mise à jour de connaissances incertaines en se basant sur l'approche de modélisation de Dempster-Shafer [Yager & al., 1994; Denœux, 2000]. L'approche de Dempster-Shafer également appelée la théorie de l'évidence, convient particulièrement aux problèmes d'apprentissage et de modélisation dans un univers de discours fermé.

Nous venons de présenter quelques méthodes ayant des capacités à prendre en compte les évolutions en environnement dynamique. Les réseaux neuronaux à structure dynamique se montrent particulièrement adaptés à la classification dynamique. Leurs principes et propriétés intéressantes seront montrés dans les sections suivantes à travers l'étude de trois algorithmes neuronaux à architecture évolutives développés pour la classification incrémentale.

1.3.2 Cluster Detection and Labeling (CDL)

Le réseau CDL a été introduit en 1998 [Eltoft, 1998]. Elaboré avec une architecture neuronale constructive, le CDL dispose de règles d'apprentissage non-supervisé lui conférant des capacités d'auto-adaptation en environnement de données non-stationnaires. Son architecture est constituée de trois couches : une couche d'entrée, une couche cachée, une couche de sortie et des connexions. Contrairement à la couche d'entrée, la couche cachée et celle de sortie sont libres d'évolution. Un réseau annexe servant à la mesure de similarité est intégré entre la couche d'entrée et la couche de sortie. Le processus d'apprentissage du CDL est élaboré en trois phases : **classification**, **fusion**, **évaluation**.

1.3.2.1 Principe du CDL

Les classes du CDL sont représentées par des prototypes hyper-sphériques. La zone d'influence de chaque prototype est décrite par une mesure de similarité évaluant le niveau de ressemblance des données par rapport aux prototypes existants. Cette mesure S est définie entre un point X_i et un prototype P_j de moyenne $\bar{X}_j$ par l'inverse de la distance euclidienne :

$$S(X_i, P_j) = \left[d(X_i, \bar{X}_j) \right]^{-1} = \left[\sqrt{\langle X_i, \bar{X}_j \rangle} \right]^{-1} \qquad 1.1$$

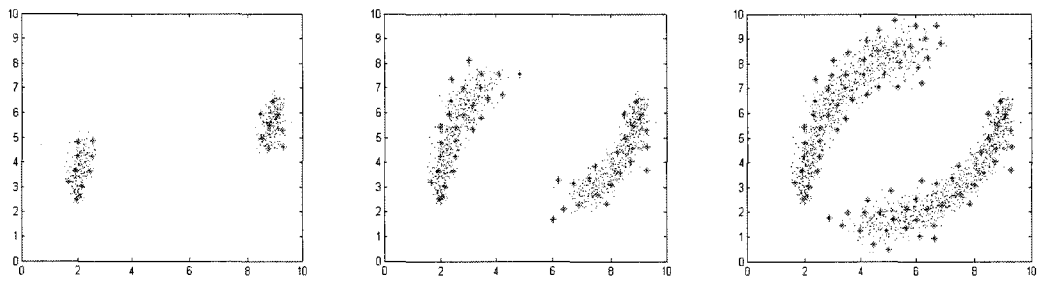

Figure 1.4 : Illustration 2D de la classification dynamique avec le réseau CDL

Dans la phase de classification du CDL, ont lieu la création des prototypes et des classes et ainsi que leur mise à jour à partir de la mesure de similarité. Les classes partageant des données ambiguës sont fusionnées dans la phase de fusion. Enfin, la phase d'évaluation permet l'élimination des classes de cardinalité très faible. La figure 1.4 présente une illustration de la classification du CDL dans un environnement de données non-stationnaires.

1.3.2.2 Discussion sur le CDL

Lors du processus de classification du CDL, plus une donnée est proche du prototype plus la mesure de similarité est grande et inversement. Le réseau CDL utilise une mesure de similarité décrivant les prototypes à l'aide d'un modèle hyper-sphérique. Une telle approximation de la structure de classes ne garantit pas une bonne qualité de classification. En effet la distribution des données est très souvent inconnue et rarement modélisable avec une forme hyper-sphérique. Par ailleurs, la mesure de similarité n'étant pas bornée, elle ne peut pas être considérée comme une fonction d'appartenance [Zadeh, 1965]. En pratique, une telle mesure met en évidence un problème de sensibilité au voisinage du centre du prototype. Les principaux avantages de ce réseau proviennent des capacités d'auto-adaptation de son architecture et de l'utilisation de règles d'apprentissage non-supervisé permettant la définition de nouveaux prototypes et/ou de classes par la création de nouveaux neurones. Malgré ces capacités, le réseau CDL ne dispose pas de règles de mise à jour servant à l'adaptation des classes évolutives dans un environnement dynamique. Néanmoins, nous verrons dans le chapitre 4, qu'en s'inspirant du principe du CDL, on peut créer un algorithme très performant pour la classification dynamique de données non-stationnaires.

1.3.3 Fuzzy Min-Max Clustering

En 1993, Simpson introduit l'algorithme neuronal FMM *Clustering* [Simpson, 1993]. Développé à partir de l'algorithme supervisé FMM Classification [Simpson, 1992], le FMM *Clustering* est doté de règles d'apprentissage non-supervisé. Le réseau FMM *Clustering* dispose d'une architecture évolutive à trois couches : une couche d'entrée, une couche cachée, une couche de sortie et des connexions. La couche d'entrée est de taille fixe tandis que la couche cachée et la couche de sortie évoluent avec le nombre de prototypes et de classes. L'apprentissage du FMM *Clustering* est mené selon un processus en trois phases : **phase d'expansion**, **test de recouvrement** entre prototypes et **phase de contraction**.

1.3.3.1 Principe du FMMC

Le FMM *Clustering* est une méthode de coalescence floue et agglomérative à base de prototypes hyper-cubiques flous. Chaque prototype P_j est défini par un vecteur de paramètres $[V_j,W_j]$ correspondant aux points Min et Max de la zone d'influence du prototype. Simpson introduit la fonction d'appartenance μ d'une donnée X_i à un prototype P_j telle que :

$$\mu(X_i, P_j) = \prod_{d=1}^{D}\left[1 - f_\alpha\left(v_j^d - x_i^d\right) - f_\alpha\left(w_j^d - x_i^d\right)\right] \text{ avec } f_\alpha(x) = \begin{cases} 1 & \text{si } \alpha.x > 1 \\ \alpha.x & \text{si } 0 \le \alpha.x \le 1 \\ 0 & \text{si } \alpha.x < 1 \end{cases} \quad 1.2$$

où x_i^d est la $d^{ème}$ composante du vecteur $X_i \in \Re^D$ et le coefficient $\alpha \in [0,1]$ règle la pente de la fonction d'appartenance de l'hyper-cubique.

L'initialisation du réseau FMM se fait avec la première donnée par la création du premier prototype et de la première classe. Lors de la phase de dilatation, les prototypes P_j sont adaptés avec les données les plus proches X_i à l'aide des règles de mise à jour suivantes :

$$v_{j,new}^d = \min\left(v_j^d, x_i^d\right) \text{ et } w_{j,new}^d = \max\left(w_j^d, x_i^d\right), \quad \forall d \in [D] \quad 1.3$$

Pour maîtriser l'expansion des prototypes, leur adaptation doit au préalable être validée par un critère d'expansion utilisant un seuil de vigilance ϑ. Après l'adaptation d'un prototype, un test de recouvrement vise à détecter d'éventuels chevauchements avec d'autres prototypes. Ceux-ci sont alors éliminés dans la phase de contraction.

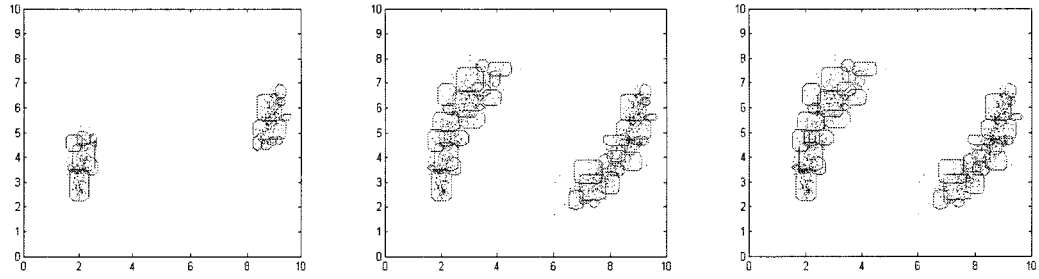

Figure 1.5 : Illustration 2D de la classification dynamique avec le réseau FMMC

1.3.3.2 Discussion sur le FMMC

Le FMM *Clustering* a la capacité de modéliser des classes par la création et l'adaptation de prototypes à partir de l'incorporation incrémentale de données. L'architecture évolutive du réseau FMM *Clustering* et son mode d'apprentissage non-supervisé confèrent à l'algorithme des aptitudes intéressantes pour la classification en environnement de données non stationnaires. Un algorithme combinant les avantages du FMM *Classification* et du FMM *Clustering* est proposé dans [Gabrys & Bargiela, 2000]. Cet algorithme qui a la possibilité de traiter à la fois des données étiquetées et non étiquetées permet en plus de prendre en compte l'incertitude dans les données. Cependant, les algorithmes de la famille FMM présentent un inconvénient majeur concernant la qualité de la modélisation. En effet, l'utilisation du modèle hyper-cubique est une approximation grossière de la distribution des données. Le réglage de la pente par le coefficient α et le seuil de vigilance ϑ ne permettent pas de résoudre ce problème. Si le coefficient α est choisi trop grand, il se pose un problème de sur-apprentissage tandis que la fixation d'un seuil de vigilance trop petit conduit à la création d'un nombre trop important de prototypes. Ces algorithmes sont dans tous les cas victimes des

problèmes d'optima locaux. En plus de ces inconvénients, la règle d'adaptation du FMMC ne permet pas le suivi de classes évolutives, elle se limite à adapter leur expansion. Noter qu'il existe d'autres algorithmes de la famille FMM. Ces derniers sont introduits pour des applications particulières. On peut citer par exemple, le FMM *Classification & Detection* [Meneganti & al., 1998] et *Modified* GFMM [Doye & Sontakke, 2002].

1.3.4 Evolving Self-Organizing Map (ESOM)

Introduit en 2003 [Deng & Kasabov, 2003], l'ESOM se présente comme une extension de l'algorithme SOM [Kohonen, 2001] pour les problèmes de classification en ligne. Basé sur les cartes auto-organisatrices de Kohonen et doté d'une architecture constructive, le réseau ESOM dispose de capacités d'apprentissage non-supervisé. Cet algorithme a une structure topologique décrite par des nœuds (ou prototypes) et des connexions dans l'espace de représentation. Le processus d'apprentissage de l'algorithme ESOM est constitué de trois phases : **compétition**, **création & adaptation**, et **élagage**.

1.3.4.1 Principe de l'ESOM

Chaque classe de l'ESOM est représentée par un ensemble de nœuds connectés formant un réseau topologique disposé en fonction des regroupements des données. Un nœud ou prototype modélise un ensemble de données suffisamment proches. Chaque nœud est caractérisé par un vecteur poids dont la taille est égale à celle de la dimension de l'espace et pondéré par une fonction d'activation a_j. Les connexions mémorisent le niveau de proximité des nœuds de la même classe et sont activées par une fonction de voisinage h_j. Le réseau ESOM est initialisé avec la première donnée par la création du premier nœud représentant à cet instant la première classe. Dans la phase de compétition, la proximité entre une nouvelle donnée X_i et les nœuds w_j du réseau est mesurée en utilisant la distance euclidienne et un seuil d'appartenance ε telle que :

$$\|w_j - X_i\| - \varepsilon > 0 \qquad 1.4$$

Si la donnée n'est pas suffisamment proche d'un nœud, un nouveau nœud est créé, sinon le nœud gagnant est adapté en utilisant la règle de mise à jour récursive avec le ratio d'apprentissage η de telle sorte que :

$$w_{j,new} = w_j + \eta h_j \left(w_j - X_i\right) \qquad 1.5$$

$h_j = a_j \Big/ \sum_{j=1}^{J} a_j$ est la fonction de voisinage avec $a_j = \exp\left(-\frac{2}{\varepsilon^2}\|w_j - X_i\|^2\right)$.

De façon périodique, le réseau est élagué en supprimant les connexions de faible poids et les nœuds inactifs. Cette phase d'élagage a pour but d'éliminer le bruit dans les données. La figure 1.6 donne une illustration de la classification du réseau ESOM.

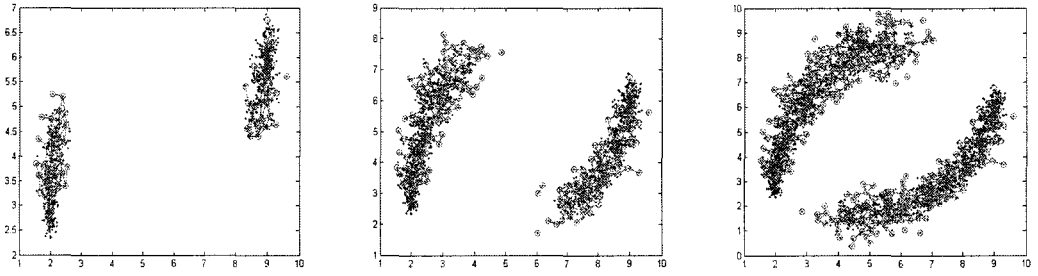

Figure 1.6 : Illustration 2D de la classification dynamique avec le réseau ESOM

1.3.4.2 Discussion sur l'ESOM

Le réseau ESOM utilise des nœuds et des connexions qui modélisent la structure des données et leur évolution dans l'espace de représentation. Dans le cadre de l'apprentissage en ligne, la procédure d'adaptation récursive des paramètres du réseau ESOM est simple et très rapide. En effet, contrairement à la plupart des algorithmes utilisant des modèles paramétriques, l'ESOM est à l'abri des problèmes de dimensionnalité [Jain & al., 1999]. Les phases d'adaptation et d'élagage de l'ESOM offrent au réseau des possibilités de fusion et scission en environnement non-stationnaire. Ce modèle est une approximation de la distribution des données définissant une zone d'influence hyper-elliptique autour de chaque nœud. Cette approximation est satisfaisante dans beaucoup de problèmes de classification. Cependant, la qualité de la classification n'est pas garantie pour des données à distributions plus complexes. Par ailleurs, la règle de mise à jour de classes de l'ESOM ne permet pas d'effectuer le suivi des classes évolutives. De même, les frontières réelles de ces classes ne sont pas connues, l'algorithme ESOM se contente de construire un réseau topologique constitué de nœuds connectés. Il existe d'autres algorithmes basés sur les cartes auto-organisatrices de Kohonen [Lapidot & al., 2002 ; Voegtlin, 2002]. Ils sont dotés de règles de mise à jour récursives mais restent limiter aux problèmes d'apprentissage non-supervisé.

Nous venons d'étudier quelques algorithmes ayant des capacités intéressantes pour la classification des données non-stationnaires. Cependant, ces algorithmes restent très limités pour résoudre la problématique de classification dynamique. Ces limitations sont essentiellement liées à leurs modèles de classification non adaptés à la distribution des données et à leurs processus d'apprentissage limités. Nous portons, par ailleurs, un grand intérêt à leurs principes de mise en œuvre notamment leurs architectures évolutives qui nous serviront de sources d'inspirations pour la suite de nos travaux. Dans la partie suivante, nous proposons une description générique de classifieurs dynamiques.

1.4 Description générique de classifieurs dynamiques

Cette description générique décrit un formalisme général pour la conception d'algorithmes de classification dynamique dans un environnement de données non-stationnaires. Très peu de tentatives existent dans la bibliographie. Comme nous l'avons vu dans la partie 1.2, cette problématique comporte des difficultés très complexes. Il s'agit notamment de problèmes de classification incrémentale, adaptative et non-supervisée auxquelles s'ajoutent la prise en compte des phénomènes dynamiques liés à l'évolution des classes. Dans ce contexte, l'objectif est donc d'effectuer une modélisation continue et adaptative des connaissances (classes) à partir des données non-stationnaires. Pour atteindre cet objectif, il est important que le classifieur dynamique soit doté de règles d'apprentissage permettant l'incorporation de nouvelles informations et la prise en compte de l'évolution des connaissances et des phénomènes dynamiques associés. Nous proposons d'établir une description générique de classifieur dynamique en considérant trois points principaux. Le premier point pose les conditions sur la technique de modélisation en vue d'une description adéquate des classes en respectant la distribution des données. Les étiquettes des données n'étant pas connues *a priori*, le second point vise à déterminer la classification (affectation) en ligne des données à l'aide d'un critère de similarité (ou de ressemblance). Enfin, le dernier point consiste à définir des règles d'auto-adaptation permettant au classifieur dynamique la prise en compte des évolutions de la partition dynamique au travers de l'intégration des nouvelles informations.

1.4.1 Modélisation de la partition dynamique

L'acte de classification consiste à grouper les données suivant leur ressemblance afin de former des groupes homogènes. Ceci se fait à travers la modélisation des connaissances utiles contenues dans les données sous forme de classes. Le modèle de classification doit décrire la distribution des données dans chaque classe et définir les frontières de décision. Avant d'étudier ce modèle, nous introduisons les hypothèses suivantes sur les données :

- Les données sont non-stationnaires, c'est-à-dire des données issues d'un processus dont le comportement est susceptible d'évoluer dans le temps. La base d'apprentissage est incomplète, elle est enrichie au fur et à mesure des acquisitions.
- A chaque instant, les données X_t de dimension D fixe, sont générées suivant une loi inconnue $P_t(X_t)$ dans χ. On suppose que : $P_{t+\Delta t}(X_t) \approx P_t(X_t)$ et $\|P_{t+\Delta t}(X_t) - P_t(X_t)\| \leq \varepsilon.\Delta t$.
- Les étiquettes des données et le nombre de classes ne sont pas connus *a priori*. L'espace de classification χ correspond à un univers de discours ouvert.

Compte tenu de ces hypothèses, l'objet de classification dynamique consiste en la double tâche de labelliser chaque nouvelle donnée et de modéliser en continu les classes susceptibles d'évoluer dans le temps. La question fondamentale et primordiale qui se pose est : quel modèle sera adéquat pour représenter en continu les connaissances utiles à extraire des données i.e. les classes ? La réponse à cette question n'est pas facile. En effet, le choix du

modèle est fortement lié à la distribution des données. Or, cette distribution n'est pas connue *a priori*. De même, pour les classes évoluant dans le temps, il est important que le modèle de classification puisse s'adapter aux évolutions de la partition dynamique. Afin d'obtenir une meilleure modélisation des classes, les conditions suivantes doivent être vérifiées :

- Le modèle de classification s'adapte correctement à la distribution de données de façon à respecter le contour et la géométrie de chaque classe.
- Le modèle de chaque classe est décrit dans une région continue et bornée de l'espace.
- Deux modèles décrivant des classes différentes ne se chevauchent pas, compte tenu de la définition 1.2.

Dans de nombreux problèmes, le modèle de classification est choisi dans la famille exponentielle [Mora, 1986] (Annexe A.1). On retrouve dans cette famille la plupart des modèles de densités présentes dans la nature. Nous étudierons dans le chapitre suivant quelques techniques de modélisation.

Nous allons décrire maintenant le formalisme introduit pour modéliser la partition dynamique des données non-stationnaires. Ensuite, on présente le critère de similarité avant de détailler le processus d'apprentissage du classifieur dynamique. Enfin, une architecture neuronale à trois couches décrit les capacités d'évolution de l'algorithme générique.

1.4.1.1 Formalisation

Considérons Π la partition dynamique de données dans l'espace χ et $\Im$ le modèle de classification associé. Supposons que chaque classe C_m est modélisable par une fonction d'apprentissage $\psi_m \subseteq \Im$ de vecteur paramètre Θ_m^t (variable temporelle). A chaque instant t, la partition dynamique $\Pi^{(t)}$ et le modèle de classification $\Im^{(t)}$ sont définis par les ensembles[1] :

$$\Pi^{(t)} = \{C_1, ..., C_m, ..., C_M\}_t \qquad 1.6$$

$$\Im^{(t)} = \{\psi_1, ..., \psi_m, ..., \psi_M\}_t \qquad 1.7$$

tels qu'on ait la relation suivante :

$$\forall X \in \chi,\ P(X \in C_m) = \alpha \ \Rightarrow\ \psi_m(X) - \mu_\alpha \geq 0 \qquad 1.8$$

où P désigne la probabilité, $\alpha, \mu_\alpha \in [0,1]$ est un coefficient réel. M est le nombre (non fixe) de classes. A partir de la relation (1.8), on peut écrire :

$$\forall (C_m, \psi_m) \in \Pi \times \Im,\ C_m = \{X \in \chi / \psi_m(X) - \mu_{\alpha_0} \geq 0\} \qquad 1.9$$

---

[1] Les notations C_m^t et ψ_m^t désignant les classes et fonctions temporelles sont remplacées par C_m et ψ_m par souci de simplicité. Nous reviendrons aux premières notations à chaque fois que cela s'avère nécessaire.

Remarque : L'équation $\psi_m(X)-\mu=0$ fixe les bornes limites (ou la frontière) de la classe C_m.

En environnement dynamique, la modélisation est souvent difficile à cause des variations plus ou moins complexes des classes évolutives. Il est donc important que la fonction d'apprentissage ψ soit assez flexible pour modéliser les contours complexes et facile à adapter pour suivre les évolutions des classes.

1.4.1.2 Modélisation optimale

La recherche du modèle de classification optimal est souvent effectuée avec une méthode de maximisation de la loi de *vraisemblance*. Dans le cadre de la classification dynamique, l'estimation du modèle de classification optimale $\Im_{opt}$ à chaque instant t se fait de façon séquentielle. Théoriquement, on peut écrire que le modèle de la partition dynamique recherché à chaque l'instant t est celui maximisant l'expression :

$$\arg\max_{\Im_{opt}^{(t)}}\left(\prod_{m=1}^{M}\prod_{i=1}^{t}P\left(X_i\in C_m^t\right)\right) \qquad 1.10$$

Une manière équivalente d'évaluer la qualité du modèle de classification est de mesurer la probabilité de mauvaise classification, c'est-à-dire, la probabilité des données classées avec erreurs. Dans ce cas, on parle de minimisation *de Risque* d'apprentissage :

$$\arg\min_{\Im_{opt}}\left(\prod_{m=1}^{M}\prod_{i=1}^{t}P\left(X_i\rightarrow C_m^t / X_i\notin C_m^t\right)\right), \qquad \rightarrow : \text{Affectation} \qquad 1.11$$

En effet, minimiser le Risque d'apprentissage d'un classifieur revient à maximiser le critère de Vraisemblance [Guermeur & Paugam-Moisy, 1999]. La difficulté de mise en œuvre de telles fonctions objectifs réside dans le fait que les données sont non-stationnaires et les classes sont évolutives. Cependant, dans la plupart des situations, l'estimation du modèle de classification en ne tenant compte que du critère (1.11) conduit à des problèmes d'optima locaux. Ces problèmes sont précisément liés aux modèles souvent mal choisis pour modéliser les classes. Une solution intéressante pour éviter ces problèmes est la régularisation [Vapnik, 1998]. C'est ainsi qu'en apprentissage séquentiel, un terme de régularisation instantanée $G(\Im^{(t)})$ est ajouté au risque. Par ailleurs, on peut exprimer un critère de convergence du classifieur dynamique :

$$\lim_{t\rightarrow\infty}\left(\prod_{m=1}^{M}\prod_{i=1}^{t}P\left(X_i\rightarrow C_m^t / X_i\notin C_m^t\right)\right)=\ell \qquad 1.12$$

avec $\ell\in\Re-\{-\infty,\infty\}$ une limite finie.

1.4.2 Critère de similarité

En classification non-supervisée, les étiquettes des données n'étant pas connues *a priori*, l'affectation des données aux classes se fait à l'aide d'un critère de similarité (de

ressemblance). Ce critère décide de la construction et de l'adaptation des classes. C'est donc un élément déterminant de la fiabilité du classifieur. La mise œuvre d'un critère de similarité passe tout d'abord par la définition d'une mesure de similarité qui a pour objectif de déterminer l'apport d'information (d'un point de vue de la ressemblance) d'une nouvelle donnée X_t par rapport aux C_m déjà créées. En classification dynamique, nous définissons cette mesure comme une fonction S telle que:

$$\begin{aligned} S:\ & \chi \times \Pi \rightarrow \Re \\ & (X_t, C_m) \rightarrow S(X_t, C_m) \end{aligned} \qquad 1.13$$

avec les conditions suivantes :

- S décroît sur $\Re$ lorsque X_t s'éloigne de C_m dans toutes les directions de χ.
- S est bornée dans un intervalle $[a,b]$ telle que : $S(X_t, C_m) = a$ si X_t est infiniment loin de la classe C_m et $S(X_t, C_m) = b$ si X_t est infiniment proche (similaire) de C_m.

Dans certaines situations, la fonction d'apprentissage ψ sert de mesure de similarité. Cependant, les bornes de ψ n'étant pas toujours connues, l'utilisation d'une telle fonction peut engendrer des problèmes de sensibilité tels que ceux constaté dans la section 1.3.2.

La notion de similarité peut s'exprimer comme la proximité géographique dans l'espace d'origine χ ou encore dans un autre espace issu d'une transformation des données. Parmi les solutions envisageables, les plus répandues sont celles basées sur des métriques (distance euclidienne, de Mahanalobis,...), des degrés d'appartenance, des fonctions noyaux, ... etc.

Enfin, le critère de similarité donne l'ensemble Π^{win} des classes gagnantes à chaque fois qu'une nouvelle donnée se présente. Ce critère est défini tel que :

$$\Pi^{win} = \{C_m / S(X_t, C_m) - \varepsilon_{th} \geq 0\} \qquad 1.14$$

où ε_{th} est le seuil de similarité. Dans certains cas, il est possible de déterminer ce seuil par le calcul, sinon il est choisi suite à différents essais.

1.4.3 Processus d'apprentissage

Le classifieur dynamique est élaboré avec un processus d'apprentissage lui conférant des aptitudes d'apprentissage en environnement des données non-stationnaires. Ce processus permet de construire et d'adapter le modèle de classification à travers l'intégration continue des nouvelles informations afin de prendre en compte les évolutions de classes. Pour permettre la prise en compte des phénomènes dynamiques décrits dans la problématique (partie 1.2 du présent chapitre), le processus d'apprentissage est décrit par quatre procédures :

- La **procédure de création** qui permet la création de nouvelles classes avec une *règle d'initialisation* de modèle.
- La **procédure d'adaptation** qui est basée sur une *règle de mise à jour* récursive et qui a pour but d'adapter les modèles de classes subissant des modifications locales et des évolutions avec glissements.
- La **procédure de fusion** qui est développée pour la fusion d'informations mutuelles, cette procédure utilise des *règles de fusion* évitant le problème de chevauchement de classes.
- La **procédure d'évaluation** qui permet la correction de certaines imperfections du modèle de classification à l'aide des *règles de scission* pour les classes à données non similaires et des *règles d'élimination* pour les classes parasites et obsolètes.

Le choix de la procédure d'apprentissage est déterminé par le critère de similarité établi par l'expression (1.14), à travers la règle de décision présentée dans le Tableau 1.1 :

Cas 1	$\text{Card}(\Pi^{win}) = 0$	Procédure de Création
Cas 2	$\text{Card}(\Pi^{win}) = 1$	Procédure d'Adaptation
Cas 3	$\text{Card}(\Pi^{win}) \geq 2$	Procédure de Fusion
Cas 4	Toutes les T données	Procédure d'Évaluation

Tableau 1.1 : Règle de décision et procédures d'apprentissage.

1.4.3.1 Procédure 1 : Création de classe, Initialisation de modèle

Lorsque la règle de décision du classifieur dynamique conduit au cas 1 (tableau 1.1), on se trouve dans l'une de deux situations suivantes :

- A l'instant initial, la première donnée $X_t = X_1$ est acquise dans un espace de classification « vide » i.e. $\Pi = \varnothing$ ($M = \text{card}(\Pi) = 0$), alors qu'aucune classe n'est encore créée.

- Une ou plusieurs classes sont déjà crées et la nouvelle donnée $X_{new} = X_t$ n'appartient à aucune des classes existantes. C'est une situation de rejet en distance.

Dans ces deux situations, le classifieur dynamique détecte l'apparition de nouveauté. Cette détection entraîne la création d'une nouvelle classe C_{new} initialisée avec l'information apportée par la donnée X_{new}. L'initialisation du modèle de classe à l'aide de l'information contenue dans une seule donnée, n'est pas toujours simple. Cette tâche est souvent rendue possible en adoptant certaines hypothèses simplificatrices sur le modèle initial ou en se basant sur l'analyse d'échantillons de données préalablement extraits. De cette façon, on peut écrire une règle d'initialisation R_{init} qui détermine le modèle initial ψ_{new} à partir de la donnée X_{new} sachant l'ensemble des hypothèses d'initialisation H_{ini} :

$$\psi_{\text{new}} = R_{init}(X_{new}/H_{ini}) \qquad 1.15$$

Après la création d'une nouvelle classe, les ensembles Π et $\Im$ sont incrémentés :

$$\Pi^{(t)} = \Pi^{(t-1)} \cup \{C_{new}\} \quad 1.16$$

$$\Im^{(t)} = \Im^{(t-1)} \cup \{\psi_{new}\} \quad 1.17$$

Cependant, comme nous l'avons vu dans la partie 1.2, la création de classe suite à l'apparition de nouveautés présente quelques fois des risques liés au bruit. Pour ne pas affecter la qualité de la modélisation, une classe créée ne sera considérée comme représentative dans la partition que si d'autres données supplémentaires arrivent dans sa zone d'influence pour développer son modèle. Sinon, c'est une classe parasite qui sera éliminée dans la Procédure 4. Une autre solution consiste à attendre d'acquérir suffisamment de données similaires avant de créer une classe. Cependant, en plus de la détection de nouveautés, cette solution requiert la mise en œuvre d'un second critère qui analyse d'une part, la similarité entre les nouvelles données et d'autre part, la cardinalité de classe qu'elles forment.

1.4.3.2 Procédure 2 : Adaptation de classes, mise à jour de modèles

Considérons cette fois que la règle de décision du classifieur conduit au cas 2 (tableau 1.1). Cela signifie donc que la nouvelle donnée X_t est suffisamment proche d'une et une seule classe C_m pour contribuer à sa définition. L'adaptation de cette classe passe par la mise à jour récursive de son modèle ψ_m avec l'incorporation de la nouvelle information. La difficulté de mise en œuvre d'une règle de mise à jour récursive est fortement liée à la nature et la complexité du modèle de classification choisi. De manière générale, nous pouvons introduire quelques conditions essentielles que doit satisfaire la règle de mise à jour R_{update} pour permettre la modélisation dynamique de données non-stationnaires :

- Adaptation du modèle aux modifications locales quelconques et suivi des évolutions de classes en environnement dynamique : la mise à jour doit se faire à travers l'ajout de nouvelles informations X_{new} et le retrait d'informations obsolètes X_{old}. La règle de mise à jour récursive R_{update} peut alors s'écrire de façon générale :

$$\psi_m^t = R_{update}\left(\psi_m^{t-1}, X_{new}, X_{old}\right) \quad 1.18$$

- Convergence de la procédure d'adaptation : supposons qu'il existe un estimateur de densité $\hat{\psi}_{est}$ donnant le modèle théorique en connaissant *a priori* l'ensemble des données de la classe $\left\{X_i / X_i \in C_m^t\right\}$, nous définissons une propriété de convergence de la règle de mise à jour récursive de la manière suivante :

$$\hat{\psi}_{est}\left[\left\{X_i / X_i \in C_m^t\right\}\right] - R_{update}\left(\psi_m^{t-1}, X_t, X_{old}\right) = \varepsilon_\psi^t \rightarrow \varepsilon, \quad \forall t \gg t_s \text{ (en temps suffisant)} \quad 1.19$$

où - est un opérateur de différence évaluant l'erreur entre deux modèles. ε est l'erreur de convergence. Si $\varepsilon = 0$, nous dirons que R_{update} est une règle de mise à jour exacte.

- Complexité moindre en temps de calcul : Le temps d'exécution de la procédure d'adaptation doit nécessairement être inférieur à celui de l'estimateur du modèle théorique. Sinon, cet estimateur, s'il existe, serait mieux adapté au problème.

Nous présentons au chapitre 3 une étude sur les méthodes de mise à jour récursives. Ces méthodes seront classées en deux groupes : les méthodes récursives exactes et les méthodes itératives basées sur le gradient stochastique.

1.4.3.3 Procédure 3 : Fusion de classes, règle de fusion

Dans le cas 3 du tableau 1, la nouvelle donnée se trouve suffisamment proche de deux ou plusieurs classes pour participer à leur définition. Ce cas révèle une situation d'ambiguïté [Dubuisson, 1990]. En effet, une donnée ne peut être affectée qu'à une seule classe, contrairement au cadre de la classification floue où l'information d'une donnée est partagée par toutes les classes. Pour traiter les problèmes d'ambiguïté, plusieurs solutions sont proposées :

Solution 1 : adapter la classe la plus proche ou celle ayant la plus grande cardinalité.

Solution 2 : adapter toutes les classes gagnantes avec la même information.

Solution 3 : fusionner les classes concernées par l'ambiguïté.

La première solution entraîne des problèmes de chevauchement de classes et ne respecte donc pas la condition de continuité de la région décrite par chaque classe. Quant à la seconde solution, en plus des inconvénients de la première, elle affecte la même donnée à plusieurs classes différentes. Ce qui est difficile à justifier dans beaucoup d'applications réelles.

La troisième alternative qui consiste en la fusion de classes semble être la meilleure solution et se justifie dans un cadre de classification en ligne. En effet, comme nous l'avons exposé dans la problématique, compte tenu que la base d'apprentissage est incomplète, l'ambiguïté se révèle comme une information mutuelle utile pour la détection de fusion de classes. Pour éviter le risque lié au bruit, un critère d'ambiguïté est utilisé. Supposons l'ensemble χ^{amb} de données ambiguës entre toutes les classes C_{win} telles que :

$$\chi^{amb} = \{X_i,\ S(X_i, C_{win}) \geq \varepsilon_{th},\ win \in [M]\} \qquad 1.20$$

La fusion des classes C_{win} ne peut être autorisée que si la condition suivante est satisfaite :

$$\text{card}\left(\{X_i,\ S(X_i, C_{win}) \geq \varepsilon_{th},\ win \in [M]\}\right) \geq N_{amb} \qquad 1.21$$

N_{amb} : seuil d'ambiguïté i.e. cardinalité minimale de l'ensemble de données ambiguës.

Cependant, compte tenu des risques d'optima locaux. La condition précédente n'est pas toujours suffisante pour autoriser la fusion surtout en cas d'utilisation de modélisation

paramétrique. Quelquefois, il est nécessaire d'utiliser un second critère pour ne pas affecter la qualité de modélisation.

En cas de détection de fusion de classes, il reste à déterminer le modèle ψ_{merg} de la classe C_{merg} résultante de la fusion. Pour ne pas affecter la rapidité, il est souhaitable de disposer d'une règle de fusion R_{merg} donnant le modèle ψ_{merg} à partir des modèles ψ_{win} :

$$\psi_{merg} = R_{merg}(\psi_{win1}, \psi_{win2}, \ldots) \quad / \quad C_{merg} = \left\{ X \in (\cup C_{win}),\ \psi_{merg}(X) - \mu \geq 0 \right\} \qquad 1.22$$

Ensuite, le modèle ψ_{merg} est adapté avec les informations mutuelles contenues dans les données ambiguës $X_i \in \chi^{amb}$ en utilisant la règle de mise à jour (1.18). A l'issue de cette fusion, les ensembles Π et $\Im$ sont mis à jour de la façon suivante :

$$\Pi^{(t)} = \left(\Pi^{(t-1)} - \left\{ \underset{win}{\cup} C_{win} \right\} \right) \cup \left\{ C_{merg} \right\} \qquad 1.23$$

$$\Im^{(t)} = \left(\Im^{(t-1)} - \left\{ \underset{win}{\cup} \psi_{win} \right\} \right) \cup \left\{ \psi_{merg} \right\} \qquad 1.24$$

La mise en œuvre de la règle de fusion (1.22) est une tâche souvent difficile selon le modèle de classification utilisé. A défaut de disposer d'une règle de fusion adéquate, deux alternatives sont proposées. La première et la plus simple consiste à utiliser l'estimateur de densité donnant le modèle théorique à partir des données connues *a priori*. La seconde alternative vise à déterminer la classe de fusion en réinitialisant son modèle et en utilisant la règle de mise à jour sur l'ensemble des données maintenant connues pour appartenir à cette classe. Le choix de l'une ou de l'autre solution n'est conseillé que si la rapidité de calcul le permet, sinon la complexité du classifieur dynamique sera trop augmentée.

1.4.3.4 Procédure 4 : Évaluation de la représentativité du modèle

Dans la plupart des problèmes réels, compte tenu du bruit dans l'information, les connaissances extraites des données non-stationnaires comportent des imperfections. De même en environnement dynamique, l'évolution des classes et l'obsolescence de l'information conduisent à la remise en cause de certaines connaissances. Il est donc important que les classes soient régulièrement analysées afin de juger de leur considération dans l'espace de classification dynamique. Cette analyse doit se faire suivant deux aspects. Le premier aspect porte sur la modélisation des classes en respectant la condition de continuité de la région décrite par chaque classe. Le second aspect porte sur la représentativité des classes en tenant compte des informations parasites et obsolètes en fonction du problème à traiter.

a) Scission de classes, règles de scission

Lorsque la région décrite par une classe présente de fortes discontinuités (par exemple deux ou plusieurs groupes de données dissimilaires ou géographiquement séparés), les informations contenues dans ces données caractérisent la présence de deux ou plusieurs classes. En effet, l'hypothèse de continuité de la région décrite par chaque classe impose que

toutes les données d'une même classe soient similaires. Dans le cas contraire, la modélisation de la classe n'est pas correcte. La solution à ce problème consiste à diviser cette classe C_{def} en scindant son modèle pour créer autant des modèles ψ_{split} qu'il y a de classes correctes C_{split}. Il est souhaitable que la règle de scission R_{split} puisse déterminer directement tous les modèles ψ_{split} à partir du modèle ψ_{def} :

$$\left\{\psi_{split} \Big/ \bigcup_{split} C_{split} = C_{def}\right\} = R_{split}\left(\psi_{def}\right) \quad / \quad C_{split} = \left\{X_i, \psi_{split}\left(X_i\right) - \mu \geq 0\right\} \qquad 1.25$$

A l'issue de cette procédure, les ensembles Π et $\Im$ sont mis à jour de la façon suivante :

$$\Pi^{(t)} = \left(\Pi^{(t-1)} - \left\{C_{def}\right\}\right) \cup \left\{\bigcup_{split} C_{split}\right\} \qquad 1.26$$

$$\Im^{(t)} = \left(\Im^{(t-1)} - \left\{\psi_{def}\right\}\right) \cup \left\{\bigcup_{split} \psi_{split}\right\} \qquad 1.27$$

Rappelons les deux difficultés majeures posées par la scission. D'une part, la mise en place d'un critère de détection de défaut de modèle est en pratique une tâche compliquée. Et d'autre part, il est difficile de déterminer de façon fiable et certaine, les modèles de nouvelles classes résultantes de la scission. En effet, le calcul par la règle (1.25) de tous les modèles des classes résultantes de la scission à partir uniquement du modèle de la seule classe de défaut n'est possible qu'en utilisant des hypothèses supplémentaires.

b) Élimination de connaissances parasites et obsolètes

Comme cela a été montré dans la problématique (partie 1.2), le bruit présent dans les données non-stationnaires peut également affecter la qualité de classification par la création des classes parasites. Le mécanisme d'élimination vise à rechercher la robustesse au bruit. Un critère de cardinalité des classes est souvent suffisant pour détecter les classes parasites C_{weak} de telle sorte que :

$$\left\{C_{weak}\right\}_{weak \in [M]} = \left\{C_m, \mathrm{card}\left(C_m\right) < N_{\min}\right\} \qquad 1.28$$

avec $N_{\min}$, la cardinalité minimale pour considérer une classe représentative. Pour éviter de supprimer à chaque fois les classes nouvellement créées, les données contenues dans les classes parasites seront repassées dans la classification après un certain nombre d'itérations. De même, cela évitera l'élimination des classes dont les données se présentent les unes après les autres de façon intermittente.

Par ailleurs, selon la dynamique d'un processus, certaines connaissances valables dans un passé plus ou moins lointain, pourront ne plus l'être à un autre moment. Ce sont des connaissances dites obsolètes qu'il faudrait également éliminer pour ne conserver que les classes représentatives. En notant C_{old} ces classes, après la procédure d'élimination, on a :

$$\Pi^{(t)} = \Pi^{(t-1)} - \left(\left\{\bigcup_{weak} C_{weak}\right\} \cup \left\{\bigcup_{old} C_{old}\right\}\right) \quad 1.29$$

$$\Im^{(t)} = \Im^{(t-1)} - \left(\left\{\bigcup_{weak} \psi_{weak}\right\} \cup \left\{\bigcup_{old} \psi_{old}\right\}\right) \quad 1.30$$

Remarque : l'obsolescence de connaissances (classes) est une notion subjective qui s'appui sur la vieillesse des informations dans les données. La décision de l'obsolescence d'une connaissance n'est possible qu'en ayant des connaissances supplémentaires sur l'application.

1.4.4 Description sous forme d'architecture neuronale

Cette section a pour but de donner une description de classifieurs dynamiques en utilisant les réseaux de neurones. En effet, les réseaux de neurones offrent des capacités d'auto-adaptation intéressantes et très utiles en classification de données non-stationnaires. Plus particulièrement, les réseaux à structure dynamique, c'est-à-dire ceux conçus avec une architecture constructive (incrémentale) et destructive (élagage) conviennent mieux pour décrire l'apprentissage en environnement dynamique. Nous décrivons le classifieur dynamique de façon générale à l'aide d'une architecture de type *feedforward* à trois couches et des connexions (Figure 1.7) :

- Une *couche d'entrée* : chaque neurone de cette couche représente une composante du vecteur : $X = [x_1,....,x_d,....,x_D]^T$. La taille de cette couche est fixée par la dimension D de l'espace.
- Une *couche cachée* : les neurones cachés mémorisent les paramètres Θ_m^t du modèle de classification. La fonction d'activation de chaque neurone caché est définie par la mesure de similarité s . La taille de cette couche évolue avec la dynamique du modèle.
- Une *couche de sortie* : les neurones de cette couche représentent les classes. Cette couche a autant de neurones qu'il y a de classes. Sa taille est donc libre d'évolution.
- Des *connexions* de poids unitaires sont établies entre la couche d'entrée et la couche cachée et des connexions mémorisant l'influence des paramètres du modèle dans chaque classe, sont établies entre la couche cachée et la couche de sortie.

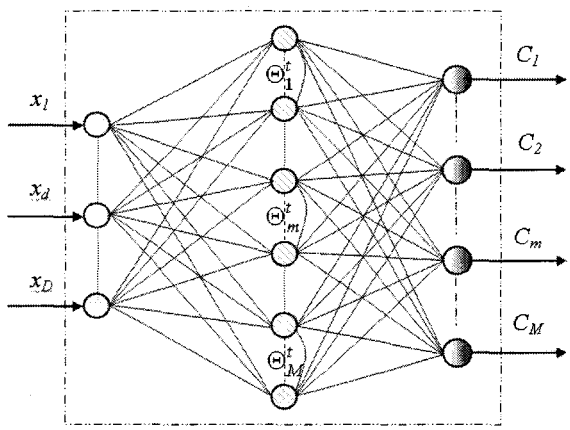

Figure 1.7 : Architecture neuronale de classifieur dynamique

Cette architecture évolue en fonction du processus d'apprentissage du classifieur dynamique. Ainsi, la création de classe se traduit par l'insertion de neurones dans la couche cachée et dans la couche de sortie : c'est la **phase constructive** de l'architecture. L'adaptation de modèle de classification entraîne la mise à jour des poids des neurones cachés et de sortie. Au cours de la procédure de fusion de classes, les neurones correspondant aux classes fusionnées sont remplacés par un neurone unique dans la couche de sortie, la couche cachée est mise à jour par la même occasion. Pendant la scission de classe, l'architecture s'adapte inversement. L'élimination de classes parasites et obsolètes entraîne la suppression de neurones dans la couche cachée et celle de sortie, c'est la **phase d'élagage** de l'architecture.

1.4.5 Algorithme générique

Pour terminer cette partie, l'algorithme générique est présenté sur la figure 1.8. Cet algorithme donne une synthèse du principe et des procédures d'apprentissage étudiées précédemment pour la mise en œuvre d'un classifieur dynamique.

Suivant cette description générique, la conception de nos classifieurs dynamiques passe par plusieurs étapes. Tout d'abord, il faut choisir un modèle de classification bien adapté pour modéliser la distribution des données. Ensuite, il est nécessaire d'établir une mesure de similarité convenable pour la classification des données non-stationnaires. Enfin, il faut déterminer les critères et règles d'apprentissage pour chaque procédure du classifieur. En fonction du modèle de classification utilisé, la mise en œuvre des règles d'apprentissage peut être plus ou moins difficile.

Algorithme générique

Choix des paramètres du classifieur dynamique
Initialisation du modèle de classification : $\Im^0$, $\Pi^0 = \varnothing$
A chaque instant t : Acquérir X_t
  Mesure de similarité : Évaluer $S(X_t, C_m^t)$
  Critère de similarité : Déterminer Π^{win}
  Cas 1 : $\text{card}(\Pi^{win}) = 0$
    Procédure de Création
  Cas 2 : $\text{card}(\Pi^{win}) = 1$
    Procédure d'Adaptation
  Cas 3 : $\text{card}(\Pi^{win}) \geq 2$
    Procédure de Fusion
  Cas 4 : Après chaque T données
    Procédure d'Évaluation
Jusqu'à l'arrêt

Figure 1.8 : Algorithme générique de classification dynamique[2]

[2] card(•) désigne la cardinalité d'un ensemble, i.e., le nombre d'éléments le constituant

On peut décliner plusieurs algorithmes de classification dynamique à partir de cette description générique. Dans le chapitre 4 (partie 4.3 et partie 4.4), on propose deux algorithmes développés suivant la démarche que nous venons d'élaborer. Le premier utilise une technique de modélisation inspirée des modèles de mélange tandis que le second modélise les classes à l'aide de fonctions d'apprentissage à base de noyau.

1.5 Conclusion

La problématique de classification dans un environnement dynamique présente des difficultés complexes et variées. Outre les problèmes de traitement en ligne des données, de nombreuses difficultés sont liées à la non-stationnarité des données. Les algorithmes se trouvent confronter aux problèmes d'apparition de nouvelles classes, de modifications locales et d'évolutions, aux phénomènes de fusion et de scission de classes, et d'élimination des informations obsolètes et parasites. En plus de ces difficultés, ces algorithmes doivent fonctionner en environnement bruité en recherchant des performances optimales et en respectant les contraintes du problème.

La plupart des algorithmes développés jusqu'à présent pour la classification des données non-stationnaires sont conçus en utilisant les techniques de réseaux de neurones pour leurs capacités à faire évoluer leurs architectures. Ces algorithmes présentent des limitations concernant principalement leurs méthodes de modélisation inadéquates lorsque la distribution des données est inconnue. En effet, l'utilisation des modèles hyper-cubiques ou hyper-sphériques pose des problèmes d'optima locaux. La qualité de modélisation se trouve ainsi fortement affectée dans des situations où les modèles de classes ne s'adaptent pas à la structure des données. En environnement non-stationnaire, le processus d'apprentissage des algorithmes neuronaux CDL, FMMC et ESOM permet la construction de classes et adapte leurs modèles aux déformations locales. Cependant, ces algorithmes n'ont pas d'aptitudes à modéliser dynamiquement les classes évolutives. De même, malgré leurs architectures constructives et incrémentales, ces algorithmes neuronaux ne sont pas aptes à prendre en compte tous les phénomènes dynamiques en environnement de données non stationnaires. La plupart des inconvénients des algorithmes précédemment exposés sont inhérents à leurs principes de mise en œuvre et à leurs techniques de modélisation.

L'analyse de la problématique et l'étude bibliographique ont permis d'orienter nos travaux de recherche. En s'inspirant des propriétés des algorithmes neuronaux à architectures évolutives, nous avons proposé une description générique de classifieurs dynamiques disposant d'un processus d'apprentissage avec des capacités d'auto-adaptation. Cette description est formalisée en 4 procédures spécifiques : création, adaptation, fusion et évaluation. La règle de décision du classifieur est basée sur un critère de similarité permettant la classification des données. Néanmoins, le développement d'algorithmes suivant cette description générique suppose que le modèle de classification soit bien choisi pour s'adapter à la distribution des données et permettre la modélisation adaptive des classes évolutives.

Afin d'atteindre ces objectifs et combler les lacunes de modélisation constatées dans les algorithmes exposés dans la partie 1.3 de ce chapitre, nous avons orienté nos recherches vers d'autres méthodes de modélisation plus performantes. C'est ainsi que le chapitre 2 présente les modèles de mélange qui sont des méthodes statistiques puissantes et les méthodes à noyau qui ont récemment montré des performances théoriques intéressantes et des résultats attractifs en classification. Dans le chapitre 3, des techniques de mise à jour récursives seront étudiées afin d'élaborer des procédures d'apprentissage incrémental et de modélisation dynamique pour les modèles de mélange et les fonctions d'apprentissage à noyau. L'ensemble des outils ainsi étudiés permettra de proposer dans le chapitre 4, deux nouveaux algorithmes dédiés à la classification dynamique de données non-stationnaires.

Chap. 2 : Modèles de mélange et Méthodes à noyau

2.1 Introduction

La partie 1.2 du chapitre 1 consacrée à la problématique de la classification dynamique, a montré les besoins de méthodes de modélisation performantes pour représenter la partition de données non-stationnaires. La plupart des modèles utilisés dans la littérature se heurte à des problèmes d'optima locaux pouvant affecter considérablement la qualité de la classification. Ces problèmes sont principalement dus à une mauvaise adéquation entre le modèle de classification et la distribution des données. Les données étant distribuées suivant une loi de densité de probabilité inconnue, définir un modèle de classification s'adaptant parfaitement à la distribution des données n'est pas une tâche facile. Le présent chapitre a pour objet la modélisation de données spatiales (non temporelles). La problématique d'évolution du modèle de classification ne sera traitée qu'ultérieurement avec l'introduction du facteur temps.

Dans le contexte de classification de données spatiales, il existe de nombreuses méthodes de modélisation pour représenter les classes. On peut les regrouper en deux grandes approches :

- L'**approche paramétrique** se base sur une approximation de la loi de densité de probabilité P supposée inconnue. L'objectif consiste à reconstituer cette loi de densité P à partir d'un ensemble de lois possibles $\{\varphi(\cdot,\Theta)\}$, Θ est le paramètre de la densité. On suppose qu'il existe un paramètre optimal Θ_{opt} tel que quelque soit la donnée X de l'espace χ, on ait $P(X)=\varphi(X,\Theta_{opt})$. De cette façon, la modélisation de classes est effectuée avec des modèles dont les éléments mathématiques sont relativement bien maîtrisés comme le modèle gaussien par exemple. En classification, il existe plusieurs méthodes d'estimation de densités paramétriques [Govaert, 2003; McLachlan & Peel, 2000]. Ces méthodes sont pour la plupart basées sur le principe du maximum de vraisemblance ou de l'estimation bayésienne. Cependant, la probabilité P étant inconnue, rien ne garantit qu'elle appartient à la famille des lois $\{\varphi(\cdot,\Theta)\}$. Pour les distributions complexes, le modèle de classification n'est pas à l'abri des problèmes d'optima locaux.

- L'**approche non-paramétrique** ne fait aucune hypothèse simplificatrice sur la distribution des données. La loi de densité de probabilité reste inconnue, on n'introduit pas de

modèle *a priori* pour représenter les classes. L'estimation non-paramétrique de la fonction de densité est effectuée à l'aide des méthodes telles que le k Plus Proches Voisins (k-PPV) [Cover & Hart, 1967], les noyaux de Parzen [Saint-Jean, 2001], ... etc. Cette approche convient à la modélisation de classes de distributions complexes. Cependant, la mise en œuvre de ces méthodes nécessite généralement l'utilisation d'une mesure de similarité (exemple : distance euclidienne) pour déterminer l'appartenance des données aux classes. En réalité, la structure des classes dépend de la nature de cette mesure, il s'agit donc d'un modèle supposé.

Dans la majorité des problèmes de classification, les méthodes statistiques se sont imposées grâce à leurs meilleurs formalismes mathématiques et les solutions intéressantes qu'elles offrent dans la plupart des problèmes. Ce chapitre est consacré à l'étude de deux méthodes statistiques : *les modèles de mélange* et *les SVM et méthodes à noyau.*

1. Les **modèles de mélange (finis)** [Govaert, 2003] : Définis avec un formalisme très général, les modèles de mélange sont des outils très performants pour la modélisation de diverses densités. L'estimation du modèle de classification optimale se fait généralement à l'aide de méthodes de maximisation du critère de vraisemblance. La plupart des travaux menés jusqu'à présent, se basent sur une approche paramétrique et apportent des solutions adéquates dans un contexte de classification automatique (*Clustering*) et non-supervisée. Il existe également des méthodes d'estimation de modèles de mélange non-paramétriques [Pilla & al., 2001]. En lien avec la problématique de notre étude, les modèles de mélanges ont connu ces dernières années quelques avancées en apprentissage séquentiel.
2. Les **SVM et méthodes à noyau** [Schölkopf & Smola, 2002] : Ces méthodes sont les plus récentes et parmi les plus exploitées actuellement en classification. A travers ces méthodes, les classes sont modélisées à l'aide de fonctions d'apprentissage à noyau. Considérées comme des modèles non-paramétriques, ces fonctions ont la capacité de représenter des formes très complexes. A partir des modèles à noyau, sont développées les techniques SVM[1] (Séparateurs à Vaste Marge) qui sont des outils puissants dont les performances en classification sont très attractives. Les SVM se basent sur le principe de minimisation de risque formalisé par Vapnik dans le cadre général de la théorie de l'apprentissage statistique. Les SVM ont d'excellentes capacités de généralisation prouvées dans [Vapnik, 1998]. Initialement utilisées pour l'apprentissage supervisé, les SVM et méthodes à noyau sont aujourd'hui utilisés dans une grande variété de problèmes allant jusqu'à la classification en ligne.

Le présent chapitre est organisé en deux grandes parties :

La première partie est consacrée aux modèles de mélange. Nous commençons par donner quelques notions essentielles sur la classification par partition. Ensuite, nous définissons les modèles de mélange dans le cadre de la classification de données spatiales et

[1] L'acronyme SVM (*Support Vector Machines*) est littéralement traduit par l'expression "Machines à Vecteurs de Support" dont le sens est très vague en français. Certains auteurs ont alors adopté la traduction de "Séparateurs à Vaste Marge" qui nous semble également plus interprétative.

décrivons le cas particulier du modèle de mélange gaussien. Après ces définitions, nous présentons le principe de maximisation du critère de vraisemblance sur lequel sont basées les méthodes d'estimation. Enfin, pour terminer cette partie, nous décrivons l'algorithme CEM (Classificatoire EM : Expectation-Maximisation) en utilisant le modèle mélange gaussien.

Dans la deuxième partie, les SVM et méthodes à noyau sont étudiés. Nous commençons par présenter le formalisme des fonctions à noyau servant à la construction de modèles d'apprentissage en classification. Ce formalisme est suivi par un exposé succinct sur la théorie d'apprentissage statistique de Vapnik. Après ces concepts théoriques de base, le principe des méthodes SVM est brièvement décrit dans le contexte de la classification. Enfin, nous détaillons la méthode d'estimation de densité à l'aide des méthodes à noyau. Quelques propriétés intéressantes de cette méthode sont illustrées à la fin de cette partie.

2.2 Modèles de mélange pour la classification

La classification des données spatiales consiste à trouver les groupements naturels des données afin de former des classes homogènes et bien séparées les unes des autres. L'homogénéité et la séparation de classes sont recherchées de telle sorte que les données d'un même groupe soient aussi similaires que possible et à l'inverse les données des groupes différents soient aussi dissemblables que possible. Les méthodes de modèles de mélange apportent un formalisme général permettant l'exploitation d'outils statistiques puissants pour la modélisation des données dans divers problèmes de Reconnaissance de Formes. A travers ces méthodes, la structure de chaque classe est décrite par une composante s'adaptant à la distribution des données; le mélange de ces composantes forme le modèle global. D'où, l'appellation de modèle de mélange. La construction de la partition optimale décrite par un modèle de mélange se fait à l'aide des techniques de classification automatique exploitant les informations contenues dans les données spatiales.

Avant de décrire les modèles de mélange, nous allons tout d'abord définir les notions de partition dure, de partition floue et de critère géométrique. Ces définitions permettent d'une part, les notions de base de modèles de mélange et d'autre part, de positionner les modèles de mélange par rapport aux autres techniques de classification (méthodes hiérarchiques).

2.2.1 Concepts de base

Les méthodes de classification couramment utilisées peuvent être classées en trois grandes familles suivant la structure de la classification produite : **hiérarchie**, **partition dure** et **partition floue**.

La classification hiérarchique consiste à déterminer une suite de partitions emboîtées depuis l'ensemble de toutes les données jusqu'aux singletons formés par chaque donnée en passant par des divisions successives. Ce type de classification, généralement basé sur des méthodes non probabilistes, présente un problème délicat quant au choix du critère d'agrégation et d'initialisation [Chelcea & al., 2004].

La plupart des méthodes de classification ont pour objectif de produire une partition. Afin de déterminer une structure de partition dure (*crisp partition*), chaque donnée est affectée à une seule classe, tandis que dans la construction d'une partition floue (*fuzzy partition*), une même donnée est partagée par toutes les classes. Plus précisément, les définitions suivantes décrivent les notions de partition dure et floue [Dang, 1998] :

Définition 2.1 (Partition dure) : Considérant un ensemble fini des données χ_L, soit $\Pi = \{C_1,...,C_M\}$ une partition constituée de M classes i.e. des sous-ensembles non vides de χ_L, Π est une partition dure si les conditions suivantes sont vérifiées :

$$\forall (C_j, C_k) \in \Pi \times \Pi, \quad C_j \cap C_k = \varnothing \qquad 2.1$$

$$C_m \in \Pi, \quad \bigcup_{m=1}^{M} C_m = \chi_L \qquad 2.2$$

La condition (2.1) traduit la propriété de disjonction de classes. La condition (2.2), quant à elle, impose que les données de toutes les classes recouvrent l'ensemble χ_L. Les valeurs d'appartenance $y_{i,m}$ des données aux classes sont binaires : $y_{i,m} = 1$ si $X_i \in C_m$ $y_{i,m} = 0$ sinon.

Définition 2.2 (Partition floue) : Considérons l'ensemble $\chi_L = \{X_1,...., X_L\}$ et μ une fonction (dite fonction d'appartenance) qui associe à tout couple (X,C) de $\chi_L \times \Pi$ une valeur $\mu(X,C)$ incluse dans $[0,1]$, une partition floue de χ_L est définie telle que :

$$\forall X_i \in \chi_L, \quad \sum_{m=1}^{M} \mu(X_i, C_m) = 1 \qquad 2.3$$

$$\forall C_m \in \Pi, \quad 0 < \sum_{i=1}^{L} \mu(X_i, C_m) < L \qquad 2.4$$

Le concept de partition floue provient de la théorie des sous-ensembles flous (*fuzzy sets*) [Zadeh, 1965] et se présente comme une extension de la notion de partition dure. La condition (2.3) traduit le fait que toute donnée appartient à la réunion de toutes les classes. La condition (2.4) signifie que toute classe contient au moins une donnée avec un degré d'appartenance non nul. De façon plus formelle, la fonction μ introduit la notion d'appartenance graduelle à un groupe. Les méthodes basées sur cette approche sont dites de **classification floue**. On peut remarquer que toute partition dure vérifie les conditions (2.3) et (2.4), on en déduit donc que l'ensemble de partitions dures est inclus au sens large dans l'ensemble des partitions floues.

La notion de partition dynamique introduite dans le chapitre 1 (définition 1.3) peut s'appliquer à une partition dure ou floue lorsque celle-ci évolue, c'est-à-dire, une partition de données non-stationnaires.

Partition optimale et critère géométrique : Pour atteindre l'objectif d'homogénéité de classes (*a posteriori*), la plupart des méthodes de classification recherche la partition optimale en optimisant un critère géométrique. Le critère le plus utilisé est la somme des inerties I de classes C_m par rapport à une donnée centrale. Il s'exprime de façon générale sous la forme :

$$Z[\Pi] = \sum_{m=1}^{M} I(C_m) = \sum_{m=1}^{M} \sum_{i=1}^{L} \mu_{i,m} \left[d(X_i, C_m) \right]^2 \qquad 2.5$$

d exprime une mesure de distance et $\mu_{i,m}$ est le degré d'appartenance de X_i à la classe C_m.

On montre que minimiser le critère (2.5) revient à minimiser l'inertie intra-classe [Dang, 1998]. Par conséquent, une méthode de classification minimisant le critère (2.5) vise la recherche de l'homogénéité de classes. Par exemple, l'algorithme de centres mobiles (*c-means*) recherche la partition optimale en minimisant le critère [McQueen, 1967] :

$$Z^{k-means}[\Pi] = \sum_{m=1}^{M} \sum_{i=1}^{L} y_{i,m} \left\| X_i - \bar{X}_m \right\|^2 = \sum_{m=1}^{M} \sum_{X_i \in C_m} \left\| X_i - \bar{X}_m \right\|^2 \qquad 2.6$$

De même, il existe une variante floue de l'algorithme de centres mobiles utilisant une fonction d'appartenance, c'est le *fuzzy c-means* (FCM) [Bezdeck, 1984]. Afin d'adapter le FCM à la modélisation de classes de formes diverses, l'algorithme générique *fuzzy c-varieties* (FCV) est proposé [Bezdek & al., 1981]. Cependant, les performances du FCV ne sont bonnes que pour des classes ayant des tailles et formes approximativement identiques. Pour pallier ce problème, le *Kernel Fuzzy C-Varieties/elliptotypes* a été récemment développé [Leski, 2004]. Cet algorithme combine le principe du FCV et les méthodes à noyau (section 2.2.3).

Le principe du *c-means* a permis de poser les bases du raisonnement sur lequel sont fondées la plupart des méthodes de classification. Le *c-means* consiste, en réalité, à déterminer un cas particulier de mélange de distributions faisant partie d'une famille plus grande appelée modèles de mélange.

2.2.2 Définition de modèles de mélange

Dans cette partie, nous allons voir que le modèle de classification d'une partition peut être déterminé à partir d'un mélange de distributions en se basant sur la modélisation probabiliste. En effet, le modèle probabiliste est le plus naturel et également le plus utilisé. Il apporte des solutions satisfaisantes aux problèmes de classification. Le principe des modèles de mélange consiste à décomposer la densité φ (*a priori* inconnue) de données en M distributions homogènes ou composantes ne se chevauchant pas ou très peu [McLachlan & Peel, 2000; Govaert, 2003]. Ces composantes correspondent aux M classes dont on cherche à estimer les paramètres $\{\Theta_m\}_{m=1,...,M}$ à partir de l'ensemble de données $\chi_L = \{X_i\}_{i=1,...,L}$. Tout d'abord, deux hypothèses sont introduites :

- Les données $X_1,...,X_L$ sont des réalisations de L vecteurs aléatoires indépendamment et identiquement distribués (i.i.d.) avec une taille L finie.
- La densité de distribution parente φ est un mélange de M distributions homogènes caractérisées par les fonctions de densité de probabilité $\{\varphi(\cdot, \Theta_m)\}_{m=1,...,M}$. Chaque composante est caractérisée par son paramètre Θ_m et sa proportion π_m dans le

mélange. Θ_m et π_m sont inconnus *a priori*. Les proportions représentent en réalité les probabilités *a priori* des classes et sont soumises aux contraintes de probabilité :

$$\forall m \in \{1,...,M\},\ 0 < \pi_m < 1 \text{ et } \sum_{m=1}^{M} \pi_m = 1 \qquad 2.7$$

La densité de probabilité $P_m(X)$ en un point X relative à une composante de paramètre Θ_m correspond à la fonction de densité $\varphi(X,\Theta_m)$ pondérée dans le mélange par la proportion π_m de la façon suivante :

$$P_m(X) = \pi_m \varphi(X,\Theta_m) \qquad 2.8$$

Le modèle de mélange est complètement décrit à partir des paramètres $\{\pi_m,\Theta_m\}_{m=1,...,M}$ de telle sorte qu'en tout point X, la densité de probabilité φ s'exprime de la manière suivante :

$$\varphi(X) = \sum_{m=1}^{M} P_m(X) = \sum_{m=1}^{M} \pi_m \varphi(X,\Theta_m) \qquad 2.9$$

Remarque : Dans le cas d'un modèle de mélange à composantes dites multimodales, chaque composante Θ_m se décompose en plusieurs composantes simples $\theta_{j,m}$ caractérisée chacune par une densité de probabilité $P_m(\bullet) = \pi_{j,m}\varphi(\bullet,\theta_{j,m})$. De cette façon, chaque composante Θ_m s'exprime alors à partir des densités des J_m composantes simples la constituant telle que :

$$\varphi(X,\Theta_m) = \sum_{j=1}^{J_m} \varphi(X,\theta_{j,m}) \qquad 2.10$$

Nous verrons au chapitre 4 (section 4.2) que le modèle de mélange multimodal est très intéressant pour la modélisation de classes complexes.

L'objectif d'une méthode de classification automatique basée sur le modèle de mélange revient donc à estimer à partir de l'ensemble $\chi_L = \{X_i\}_{i=1,...,L}$, tous les couples $(\pi_m,\Theta_m)_{m=1,...,M}$ caractérisant les composantes du mélange. Cependant, cette estimation n'a de sens que si le modèle est identifiable [Saint-Jean, 2001].

Définition 2.3 : Une famille de densité $\varphi(\bullet,\Theta)$ définie sur un modèle de mélange fini est dite identifiable si pour deux composantes distinctes Θ_1 et Θ_2, la relation suivante est vérifiée :

$$\Theta_1 = \Theta_2 \quad \Leftrightarrow \quad \varphi(\bullet,\Theta_1) = \varphi(\bullet,\Theta_2) \qquad 2.11$$

Cette définition ne s'étend pas au cas de modèles de mélange à composantes multimodales (i.e. plusieurs densités par composante). En effet, en considérant les composantes multimodales $\Theta_1 = [\pi_a,\pi_b,\theta_a,\theta_b]^T$ et $\Theta_2 = [\pi_b,\pi_a,\theta_b,\theta_a]^T$ d'un même modèle avec θ_a et θ_b des composantes simples, on a $\Theta_1 \neq \Theta_2$ mais $\varphi(\bullet,\Theta_1) = \varphi(\bullet,\Theta_2)$ à cause de la permutation des

paramètres [Stephens, 2000a]. La condition d'identifiabilité est intéressante car elle impose l'unité de la décomposition d'un modèle de mélange. Un exemple de mélange fini non identifiable est la densité uniforme pour laquelle, sur un même intervalle, on obtient une infinité de décomposition [Hogg & Whittinghill, 2001].

2.2.3 Le modèle de mélange gaussien

Dans beaucoup de problèmes de classification, en absence de connaissances particulières sur la distribution des données, les densités gaussiennes sont couramment utilisées. Cela signifie que l'on fait l'hypothèse selon laquelle les données sont générées suivant une loi normale mono ou multidimensionnelle dans un espace de dimension D. Cette hypothèse introduit une approximation justifiée dans la plupart des applications. En effet, les densités gaussiennes sont répandues dans la nature et permettent de modéliser un grand nombre de phénomènes aléatoires. Par exemple, la taille d'une personne, la pluviométrie annuelle sur une région et bien d'autres grandeurs sont des variables aléatoires centrées autour d'une valeur moyenne et distribuées avec une certaine dispersion : elles suivent donc la loi normale (ou gaussienne). Par ailleurs, l'avantage d'utiliser ces densités gaussiennes est que leurs modèles sont bien maîtrisés et l'on dispose de procédures efficaces pour les manipuler. De plus, il est intéressant de constater que la famille de densité définie par le modèle de mélange gaussien (à composantes simples) est identifiable (définition 2.3).

Dans le cas d'un modèle de mélange gaussien à composantes simples, le paramètre Θ_m de chaque composante du mélange est caractérisée par sa moyenne $\bar{X}_m$ et sa matrice de covariance Σ_m. La densité de probabilité multidimensionnelle φ, estimée conditionnellement à la classe de paramètre Θ_m, est définie en tout point X par la formule suivante :

$$\varphi(X,\Theta_m)=\frac{1}{(2\pi)^{1/D}(\det\Sigma_m)^{1/2}}\exp\left(-\frac{1}{2}(X-\bar{X}_m)\Sigma_m^{-1}(X-\bar{X}_m)^T\right) \qquad 2.12$$

De cette façon, la densité du modèle global s'exprime au point X en pondérant chaque composante par la probabilité a priori π_m de telle sorte que :

$$\varphi(X)=\sum_{m=1}^{M}\frac{\pi_m}{(2\pi)^{1/D}(\det\Sigma_m)^{1/2}}\exp\left(-\frac{1}{2}(X-\bar{X}_m)\Sigma_m^{-1}(X-\bar{X}_m)^T\right) \qquad 2.13$$

A travers les modèles de mélange, nous venons de voir comment on peut construire un modèle de classification. Lorsque les données sont connues *a priori*, il existe plusieurs méthodes capables d'estimer le modèle optimal. Ces méthodes recherchent pour la plupart la maximisation d'un critère de vraisemblance. Étudions d'abord le principe de maximisation de la vraisemblance avec l'algorithme très connu EM (Expectation-Maximisation) avant de présenter sa variante CEM (classificatoire EM) pour le mélange gaussien.

2.2.3.1 Critère de vraisemblance

Dans le cadre de la modélisation probabiliste, l'estimateur du modèle de classification $\Im$ recherche la partition optimale de manière à former des classes homogènes à partir des données $\chi_L = \{X_i\}_{i=1,...,L}$. Cet objectif se traduit par la maximisation de la probabilité P d'appartenance des données aux classes. C'est ainsi que la **loi de vraisemblance**, initialement introduite dans [Hartlet, 1958], est formulée dans le cadre général par :

$$L(\Im) = P(\chi_L, \Im) = P(\{X_1, ..., X_L\}, \Im) \qquad 2.14$$

Dans le cas où le modèle $\Im$ est un modèle de mélange, on cherche à estimer l'ensemble de couples $(\pi_m, \Theta_m)_{m=1,...,M}$ maximisant la loi de vraisemblance. Or, l'expression développée de ce critère s'écrit par le produit de densité de probabilités sur l'ensemble des données relativement à chacune des composantes Θ_m du mélange de telle sorte que :

$$L[\Im] = \prod_{i=1}^{L} \varphi(X_i) = \prod_{i=1}^{L} \sum_{m=1}^{M} \pi_m \varphi(X_i, \Theta_m) \qquad 2.15$$

Maximiser cette expression revient à maximiser la **log-vraisemblance** exprimée comme suit :

$$\log L[\Im] = \sum_{i=1}^{L} \log \sum_{m=1}^{M} \pi_m \varphi(X_i, \Theta_m) \qquad 2.16$$

L'estimation du modèle optimal $\Im_{opt}$ par la méthode du maximum de vraisemblance revient donc à trouver les paramètres Θ_m annulant la dérivée de l'expression (2.16) :

$$\Im_{opt} = \arg\max_{\Im} (\log L[\Im]) \quad \Rightarrow \quad \frac{\partial \log L[\Im_{opt}]}{\partial \Im_{opt}} = 0 \qquad 2.17$$

La relation précédente n'étant pas une équivalence, le principe de maximisation de la vraisemblance par la technique de dérivation ne garantit donc pas l'obtention d'un optimum global. Généralement, on se contente d'un optimum local qui se trouve être satisfaisant dans la plupart des problèmes. Cependant, il convient de constater que la dérivation de la log-vraisemblance est difficile à réaliser directement, voir même impossible dans certains cas.

- **Maximisation de la vraisemblance par l'algorithme EM**

L'EM (Expectation-Maximisation) est l'algorithme le plus connu parmi les méthodes d'estimation des paramètres de modèle de mélange par la méthode de maximisation de la vraisemblance [Dempster & al. 1977, Bilmes 1998]. L'algorithme EM a été développé pour la classification non-supervisée. Dans le cadre général, il détermine une partition floue de données. L'idée de cet algorithme consiste à introduire les informations manquantes (labels) sous forme de variables (inconnues) qu'il faut également estimer en même temps que les paramètres du modèle. Cela augmente certes le nombre de grandeurs à déterminer mais présente l'avantage de faciliter le processus d'optimisation du critère de vraisemblance et de déterminer les classes d'appartenance des données. A chaque donnée X_i, est associée une

grandeur $\mu_i = [\mu_{i,1},...,\mu_{i,M}]^T$ qui correspond au vecteur d'appartenance de la donnée aux classes. Les variables $\mu_{i,m}$ sont incluses dans l'intervalle [0,1] pour une description de partition floue (section 2.2.1). Avec ces informations manquantes, la maximisation de la vraisemblance se fait de façon plus pratique, par la maximisation de la **log-vraisemblance complétée** estimée sur le modèle de classification $\Im$ de paramètres $(\pi_m, \Theta_m)_{m=1,...,M}$:

$$\log L_C[\Im] = \sum_{m=1}^{M}\sum_{i=1}^{L} \mu_{i,m} \log(\pi_m \varphi(X_i, \Theta_m)) \qquad 2.18$$

L'algorithme EM recherche le maximum de vraisemblance de façon itérative. A chaque itération on a deux étapes :

- L'étape E (Expectation) qui consiste à calculer les probabilités *a posteriori* sachant les données observées. L'algorithme détermine l'espérance conditionnelle de la vraisemblance relativement au modèle existant.
- L'étape M (Maximisation), quant à elle, maximise l'espérance de la vraisemblance. Dans cette étape, l'algorithme estime les paramètres du modèle.

L'EM converge vers un optimum (global ou local) du modèle au fur et à mesure des itérations [Saint-Jean, 2001]. Il existe plusieurs variantes de l'EM : Component-Wise EM [Celeux & al., 1999], Split & Merge EM [Ueda & al., 2000], Credal EM [Vannoorenberghe & Smets, 2005]. Il existe aussi une variante de l'EM qui estime les paramètres du modèle au sens de la partition dure. C'est l'algorithme CEM dont le principe est exposé dans la section suivante.

2.2.3.2 Algorithme CEM pour le mélange gaussien

L'algorithme CEM (Classificatoire EM) a été proposé dans [Celeux & Govaert, 1995]. Il a la même structure que l'algorithme EM avec l'ajout d'une étape supplémentaire (Étape C) entre les étapes E et M. L'étape C a pour but la détermination de la classification des données au sens de la partition dure. Chaque donnée est ainsi affectée à une et une seule classe de telle sorte que les informations manquantes soient des variables binaires $y_{i,m} \in \{0,1\}$. Le critère de log-vraisemblance complétée (2.18) s'exprime donc pour l'algorithme CEM sous la forme :

$$\log L_C[\Im] = \sum_{m=1}^{M}\sum_{X_i \in C_m} \log(\pi_m \varphi(X_i, \Theta_m)) \qquad 2.19$$

Cette formulation du critère de vraisemblance est appelée **log-vraisemblance classifiante**.

Dans le cas de mélanges gaussiens, l'optimisation du critère de vraisemblance classifiante est effectuée en maximisant l'espérance $Q(\Im, \Im^{(k)})$ définie à chaque itération k par :

$$Q(\Im, \Im^{(k)}) = \sum_{m=1}^{M}\sum_{X_i \in C_m} \left(\log \pi_m^k - \frac{1}{2}\log\left(\det \Sigma_m^k\right) - \frac{1}{2}\left(X_i - \bar{X}_m^k\right)\Sigma_m^{k\,-1}\left(X_i - \bar{X}_m^k\right)^T \right) \qquad 2.20$$

Le principe du CEM pour les mélanges gaussiens est donné sur la figure 2.1.

Algorithme CEM : Mélanges gaussiens

Initialisation : $\Im^{(0)}$ et fixer ε

Répéter (à chaque itération k)

Etape E : Estimation : Calcul des probabilités *a posteriori*

$$y_{i,m}^k = \frac{\pi_m^k \varphi(X_i, \Theta_m^k)}{\sum_{i=1}^{M} \pi_m^k \varphi(X_i, \Theta_m^k)}$$

Etape C : Classification : Affectation des données aux classes

$$\begin{cases} y_{i,m}^k = 1 & \text{si } y_{i,m}^k = \arg\max y_{i,m}^k \text{ i.e. } (X_i \in C_m) \\ y_{i,m}^k = 0 & \text{sinon} \qquad\qquad\qquad \text{i.e. } (X_i \notin C_m) \end{cases}$$

Etape M : Maximisation : Calcul des paramètres de la classe

$$\pi_m^k = \frac{\text{card}(C_m^k)}{L} \quad \text{avec} \quad \text{card}(\bullet) : \text{cardinalité}$$

$$\bar{X}_m^k = \frac{1}{\text{card}(C_m^k)} \sum_{X_i \in C_m} X_i$$

$$\Sigma_m^k = \frac{1}{\text{card}(C_m^k) - 1} \sum_{X_i \in C_m} \left(X_i - X_m^k\right)\left(X_i - X_m^k\right)^T$$

Jusqu'à : $\left|Q(\Im, \Im^{(k)}) - Q(\Im, \Im^{(k-1)})\right| < \varepsilon$

Algorithme 2.1 : CEM pour le mélange gaussien

La figure 2.1 montre une illustration de la classification avec l'algorithme CEM sur un mélange gaussien constitué de 8 classes statiques dont chacune est constituée de 200 données.

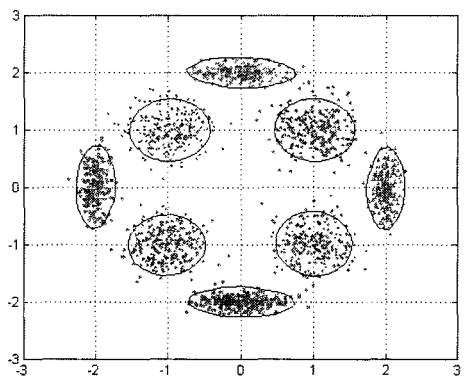

Figure 2.1 Illustration de la classification CEM avec 8 classes gaussiennes.

Remarque : En considérant un mélange gaussien à proportions toutes égales et à composantes hyper-sphériques c'est-à-dire : $\pi_m = \frac{1}{M}$ et $\Theta_m = \left(\bar{X}_m, \Sigma_m = I_D\right)$, l'algorithme CEM correspond, dans ces conditions, à l'algorithme *c-means*.

L'algorithme EM et sa variante CEM que nous venons de présenter fonctionnent de façon itérative, mais ce sont des algorithmes développés pour la classification non-supervisée

et hors-ligne. Ces algorithmes sont très gourmands en temps de calcul surtout lorsque la quantité des données est importante. Il est quelquefois possible d'introduire des hypothèses simplificatrices visant à accroître la rapidité de traitement. C'est l'objet de la section suivante.

2.2.3.3 Mélanges gaussiens parcimonieux

L'intérêt des mélanges gaussiens parcimonieux est de fournir des solutions simples pour accélérer l'estimation des paramètres du modèle par la maximisation de vraisemblance. C'est ainsi qu'en adoptant l'hypothèse simplificatrice de proportions identiques pour toutes les composantes du mélange, on facilite l'estimation des probabilités *a priori* π_m. De cette façon, on affranchit l'algorithme du calcul des π_m à chaque itération. En effet, cette hypothèse traduit une situation d'équiprobabilité et donc : $\pi_m = \frac{1}{M}$. $\forall m \in [M]$. Dans le Chapitre 5, un modèle de mélange gaussien parcimonieux sera introduit en utilisant une transformation de l'équiprobabilité en possibilités unitaires. La modélisation basée sur l'approche possibiliste est mieux adaptée aux problèmes de classification dans un univers ouvert où la modélisation probabiliste trouve ses limites, notamment à cause de la contrainte de probabilité $\sum_m \pi_m = 1$.

Par ailleurs, Celeux et Govaert [1995] montrent qu'en étudiant les caractéristiques des paramètres du modèle de mélange gaussien, il est possible de simplifier le calcul de maximisation de la vraisemblance. Ceci passe par une décomposition de la matrice de covariance qui permet d'identifier la géométrie des classes (forme, volume, orientation). C'est ainsi que dans un problème de surveillance de systèmes, en remarquant que les classes sont de forme sphérique (variable), Samé [2005] accélère l'algorithme CEM en ne déterminant à chaque itération que la valeur propre pour chaque matrice de covariance du modèle. Cependant, ces situations de simplification sont rares, le problème de la complexité n'est pas résolu dans la plupart des applications.

2.2.4 Discussion sur les modèles de mélange

Le modèle de mélange apporte un formalisme intéressant qui permet de représenter la distribution des données par une loi de densité de probabilité. Cette loi est caractérisée par autant de composantes qu'il y a de classes. Cependant, le modèle de densité doit être choisi *a priori* et le même pour toutes les classes. L'estimation des paramètres du modèle de mélange caractérisant la partition optimale passe par la maximisation d'un critère de vraisemblance. Ce critère est établi pour rechercher l'homogénéité des classes de la partition. Il existe plusieurs méthodes de maximisation de la vraisemblance développée pour un contexte de classification non-supervisée. L'algorithme EM est le plus connu et le plus utilisé. L'algorithme EM converge, mais l'optimum trouvé dépend de l'initialisation. Un autre inconvénient de ces méthodes d'estimation est leur temps de calcul trop important pour les échantillons de grande taille. Les diverses variantes de l'EM développées, les unes pour augmenter la rapidité de l'algorithme et les autres pour améliorer sa qualité de modélisation, se limitent pour la plupart à une utilisation hors-ligne.

Malgré qu'ils soient développés avec des procédures itératives, l'algorithme EM et sa variante CEM ne sont pas conçus pour les applications en ligne. En effet, pour permettre leur utilisation, il est nécessaire que les données soient complètement disponibles. Par ailleurs, ces algorithmes ne disposent pas de règles de modélisation adaptative nécessaires pour résoudre les problèmes de classification dynamique. Les récents travaux réalisés sur les modèles de mélange ont permis de développer des versions incrémentales des algorithmes EM et CEM. La version incrémentale du CEM sera exposée dans le chapitre 3.

2.3 Méthodes à noyau pour la classification

Dans la partie 2.2 de ce chapitre, quelques méthodes de classification basées sur une modélisation probabiliste de modèles de mélange ont été présentées. Ces méthodes recherchent la classification optimale en maximisant le critère de vraisemblance, critère quantifiant la qualité de modélisation à travers la recherche de l'homogénéité des classes. Contrairement à ces méthodes, les SVM et méthodes à noyau qui font l'objet de l'étude de la présente partie, recherchent la meilleure classification en minimisant un risque d'apprentissage. Cette approche se base sur la théorie d'apprentissage statistique formalisée par Vapnik [1998]. Elle est différente de l'approche bayésienne. En effet, l'approche bayésienne se base généralement sur un modèle paramétrique décrit par une densité de probabilité qui est prise en compte par le système d'apprentissage comme une connaissance *a priori*. Les SVM et méthodes à noyau sont capables de modéliser des formes complexes sans *a priori* sur la distribution des données. Ils ont connu ces dernières années une explosion considérable dans tous les domaines de l'apprentissage artificiel [Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2002]. Ces méthodes ont des fondements théoriques solides et offrent des résultats expérimentaux attractifs dans la plupart des problèmes de classification.

Dans un premier temps, nous exposons quelques généralités en décrivant tout d'abord les bases théoriques de l'apprentissage statistique, ensuite le formalisme de l'espace de Hilbert et du noyau reproduisant, et enfin la mise en œuvre des techniques SVM pour la classification. Dans un deuxième temps, nous étudions plus en détails l'estimateur de densité à l'aide des méthodes à noyau et ses propriétés intéressantes pour la modélisation de classes.

2.3.1 Généralités

2.3.1.1 Apprentissage statistique et Régularisation

La théorie de l'apprentissage statistique (*SLT : Statistical Learning Theory*) a été introduite par Vapnik [1998; 1999] dans le cadre très général de la Reconnaissance de Formes (RdF). Cette théorie a permis de formaliser le problème de l'apprentissage statistique, sous forme d'un problème de minimisation de risque. Considérons les hypothèses suivantes :

- Soit $\chi_L = \{X_1, ..., X_L\}$ un ensemble de données indépendamment distribuées suivant une fonction de probabilité $P(X)$ fixe dans l'espace χ mais inconnue.
- Supposons que l'on dispose d'un superviseur (exemple : expert humain) capable de déterminer pour chaque donnée observée $X_i \in \chi_L$, l'étiquette désirée (ou label) $y_i \in Y$ relativement à la loi de densité $P(y, X)$ également fixe et inconnue.
- Supposons qu'il existe une méthode d'apprentissage capable de déterminer un modèle F_ω, élément d'une classe de fonctions d'apprentissage $\Lambda = \{F_\omega\}$ de paramètre ω.

Vapnik décrit l'apprentissage statistique comme un problème qui consiste à choisir la meilleure fonction F_ω de Λ, capable de prédire la réponse du superviseur de la manière la plus correcte possible [Vapnik, 1999].

Formulation du problème : L'apprentissage statistique est formalisé en utilisant une fonction coût $\xi(F_\omega, X)$ qui quantifie l'erreur d'apprentissage de F_ω pour chaque donnée X par rapport à la valeur désirée y. Un exemple de fonction coût est : $\xi(F_\omega, X) = |y - F_\omega(X)|$. L'objectif de la méthode d'apprentissage consiste donc à choisir parmi toutes les fonctions F_ω de Λ, celle qui minimise l'intégrale des erreurs $\xi(F_\omega, X, y)$ sur χ. Ce principe se traduit par la minimisation du **risque structurel** défini dans le cas des variables continues comme suit :

$$R[F_\varpi, \chi] = \int_{X \in \chi} \xi(F_\varpi, X) dP(X, y) \qquad 2.21$$

Cependant, l'exploitation de cette écriture est très difficile car la densité de probabilité $P(y, X)$ est inconnue. C'est ainsi que le **risque empirique** a été défini tel que :

$$R_{emp}[F_\omega, \chi_L] = \frac{1}{L} \sum_{i=1}^{L} \xi(F_\omega, X_i) \qquad 2.22$$

Sous la condition que F_ϖ soit la meilleure fonction d'apprentissage sur χ_L, le risque structurel (2.21) est borné par le risque empirique (2.22) augmenté d'une grandeur qui dépend de la VC-dimension[2] [Vapnik, 1999]. De même, la différence entre ces deux risques converge vers 0 [Guermeur & Paugan-Moisy, 1999; Bousquet & al., 2004]. Cela signifie que lorsque le nombre de données est très important, minimiser le risque empirique revient à minimiser le risque structurel. Cependant, le principe de minimisation du risque empirique ne fonctionne que si la classe de la fonction d'apprentissage est convenablement choisie. Par exemple, si les données sont générées suivant un modèle quadratique, la meilleure fonction linéaire conduit à un sous-apprentissage (*underfitting*), et à l'opposé, une fonction (polynôme) de degré trop élevé entraîne un sur-apprentissage (*overfitting*). Il est donc nécessaire de trouver un compromis entre la distribution des données et les capacités de généralisation de la fonction d'apprentissage.

Régularisation : Pour résoudre le problème lié au principe de minimisation du risque empirique, la technique de régularisation offre une solution intéressante en limitant le choix de la fonction d'apprentissage dans un sous-ensemble adéquat. Cette solution proposée dans [Tikhonov & Arsenin, 1977; Evgeniou & al., 2000] consiste à pénaliser le critère de minimisation de risque avec un terme $G(F_\omega)$ dit de régularisation. De cette façon, l'estimation de la meilleure fonction F_ω se fait à travers la minimisation du **risque régularisé** établi de la manière suivante :

[2] La VC-dimension est une mesure de capacité de classifieur. Elle est définie dans [Vapnik, 1998]

$$R_{reg}[F_\omega, \chi_L] = R_{emp}[F_\omega, \chi_L] + G[F_\omega] \qquad 2.23$$

L'utilisation de ce risque permet de garantir de meilleures performances de généralisation en limitant d'une part, les problèmes d'optima locaux, et d'autre part, en évitant le choix de fonctions d'apprentissage trop complexes. Le choix du terme de régularisation $G[F_\omega]$ dépend du problème à traiter. En classification, la minimisation du terme de régularisation se traduit généralement par la maximisation d'une marge géométrique (voir section 2.3.2).

Le principe de minimisation du risque formulé ci-dessus (2.23), convient particulièrement aux problèmes d'apprentissage supervisé. En effet, l'utilisation d'une fonction coût nécessite dans la plupart des cas la connaissance *a priori* des sorties réelles (labels). Quelques exemples de fonctions coût dont le choix dépend naturellement du problème à traiter, sont présentés dans [Portera, 2005].

2.3.1.2 Espace de Hilbert et Noyau Reproduisant

La fonction noyau noté $\kappa(\bullet,\bullet)$ est une mesure de similarité symétrique et définie positive exprimée par le produit scalaire. Ce produit scalaire est défini dans un espace de Hilbert[3] Γ afin de conférer aux fonctions d'apprentissage à noyau des capacités de modélisation des formes complexes (non-linéaires). L'espace Γ est issu de la projection des données X_i de l'espace χ à partir d'une transformation $\phi : \chi \to \Gamma$ définie telle que $\phi(X) = \kappa(\bullet, X)$. Les fonctions d'apprentissage à noyau reposent sur une structure de représentation vectorielle appelé espace de Hilbert à noyau reproduisant *(RKHS : Reproducing Kernel Hilbert Space)*.

Définition 2.4 (RKHS) : Un *espace de Hilbert à noyau reproduisant* Γ est un espace de Hilbert de fonctions $f : \chi \to \Re$, s'il existe un noyau $\kappa : \chi \times \chi \to \Re$ de telle sorte que la propriété suivante soit vérifiée[4] :

$$\langle f(\bullet), \kappa(\bullet, X) \rangle_\Gamma = f(X), \ \forall X \in \chi \qquad 2.24$$

Cette propriété est dite de **noyau reproduisant**. En particulier, on a :

$$\langle \phi(X_1), \phi(X_2) \rangle_\Gamma = \langle \kappa(\bullet, X_1), \kappa(\bullet, X_2) \rangle_\Gamma = \kappa(X_1, X_2) \qquad 2.25$$

En considérant l'ensemble de données $\chi_L = \{X_1, ..., X_L\}$ et $\kappa(\bullet,\bullet)$ un noyau défini positif, une fonction d'apprentissage à noyau f est définie dans l'espace RKHS sous la forme générale suivante :

$$f(\bullet) = \sum_{i=1}^{L} \langle \phi(\bullet), \phi(X_i) \rangle_\Gamma + b = \sum_{i=1}^{L} \alpha_i \kappa(\bullet, X_i) + b, \qquad \alpha_i \in \Re, \ b \in \Re \qquad 2.26$$

---

[3] Les définitions d'espace de Hilbert et de noyau reproduisant (i.e. noyau défini positif) sont présentées en Annexe A.2.

[4] $\langle \bullet, \bullet \rangle_\Gamma$ est le produit scalaire dans l'espace de Hilbert Γ.

Cette fonction d'apprentissage à noyau est un modèle non-paramétrique capable d'approximer des fonctions complexes en classification [Burges, 1998; Herbrich, 2002] et en régression [Smola & Schölkopf, 2004]. Ses principaux avantages découlent de la propriété du noyau reproduisant dans l'espace RKHS. Ces avantages sont essentiellement :

- L'astuce de substitution du noyau (*kernel trick*) qui permet de traiter les non-linéarités en remplaçant le noyau produit scalaire par un noyau convenablement choisi (ex : noyau gaussien). En effet, f est linéaire en $\phi(X)$, mais non-linéaire en X.
- La simplification des calculs à travers l'utilisation des outils de l'algèbre linéaire et de la géométrie analytique rendue possible par la projection ϕ des données de l'espace d'entrées χ vers un espace de Hilbert Γ doté du produit scalaire.
- Les diverses possibilités du choix du noyau $\kappa(\cdot,\cdot)$ qui offrent des choix très variés de la mesure de similarité entre les données et donnent des configurations géométriques intéressantes (quelquefois interprétables) de la structure de l'espace RKHS.

Quelques exemples de noyaux les plus utilisés et de techniques simples servant à la construction de nouveaux noyaux définis positifs sont proposés dans [Schölkopf & Smola, 2002].

Les techniques SVM, aujourd'hui très utilisées pour l'estimation de modèles de classification, sont fondées sur le formalisme de fonctions noyau et le principe de minimisation de risque d'apprentissage.

2.3.1.3 Techniques SVM pour la classification

Les Séparateurs à Vaste Marge (SVM) ont été proposés dans les années 90 [Vapnik, 1995]. L'idée des SVM a été initialement introduite pour la recherche d'hyperplans de séparation dans le cadre de la classification binaire. Un hyperplan est caractérisé par la fonction d'apprentissage à noyau dans la formule (2.26) et fixe la frontière entre les classes dont les données sont connues *a priori*. La recherche de l'hyperplan optimal est effectuée suivant le principe de minimisation du risque régularisé (2.23) où le terme de régularisation est inversement proportionnel à la marge géométrique (entre les deux classes) mesurée dans l'espace RKSH [Burges, 1998]. Dans le cas de classes à frontière non-linéaire, l'astuce de substitution de noyau permet de trouver une fonction non-linéaire convenable pour séparer ces classes dans l'espace χ [Bennett & Campbel, 2000]. Par ailleurs, lorsque les classes sont non-séparables, une version SVM dite à marge douce (*soft margin*) est proposée dans [Schölkopf & al. 2000; Chen & al., 2005]. L'idée consiste à relâcher les contraintes afin d'autoriser la classification d'un certain nombre de données (*outliers*) avec des erreurs de relâchement (*slack errors*). La quantité d'erreurs de classification à minimiser correspond au risque empirique donné dans la formule (2.23).

D'abord développées pour la classification binaire (Annexe A.2, section A.2.2), les techniques SVM ont été très vite étendues aux problèmes multi-classes. Pour ce faire, deux approches principales existent :

1. La première approche consiste à combiner plusieurs classifieurs SVM binaires : One vs. One SVM classifier [Friedman, 1996], Ones vs. Rest SVM classifier [Weston & Watkins, 1999], Binary Tree SVM classifier [Salomon, 2001].
2. La seconde approche reformule le problème d'optimisation dans un contexte multi-classe avec la prise en compte de tous les hyperplans de séparation entre les différentes classes [Guermeur, 2002; Borer, 2003].

La plupart des techniques SVM développées pour la classification est basée sur l'approche de séparation. Pour des informations supplémentaires, nous invitons le lecteur à consulter [Burges, 1998; Schölkopf & Smola, 2002; Chen & al., 2005]. Dans ce travail, nous nous intéressons particulièrement aux méthodes basées sur une approche par modélisation visant à déterminer un modèle fidèle de chaque classe à l'aide de fonctions de densité. La section suivante est consacrée à l'étude de l'estimateur de densité à l'aide de méthodes à noyau.

2.3.2 Estimation de densité à l'aide de méthodes à noyau

L'estimateur de densité a été développé dans [Tax & Duin, 1999; Schölkopf & al. 2001] pour la classification mono-classe (sans label). Cette méthode est connue sous plusieurs appellations : SVM mono-classe (One-class-SVM), support de distribution, ensemble de niveau, détection de nouveauté (*novelty detection*), etc ... L'estimateur de densité à l'aide des méthodes à noyau se distingue des autres estimateurs de densité par trois aspects principaux :

- Il n'introduit aucun *a priori* sur la géométrie de la classe. L'estimation de densité se fait à l'aide d'une fonction d'apprentissage à noyau qui correspond à un modèle non-paramétrique capable de modéliser des classes de formes complexes.
- Contrairement aux estimateurs classiques de densité qui construisent leurs modèles sur toutes les données, la fonction d'apprentissage à noyau est définie uniquement avec les vecteurs support qui ne sont qu'un sous-ensemble restreint de la base d'apprentissage.
- L'estimation de la fonction d'apprentissage à noyau se fait à travers la détermination d'un hyperplan dans l'espace RKHS issu de la transformation des données de χ.

Tout d'abord, nous commençons par étudier la problématique d'estimation de densité comme un problème de détermination de contour de classe. Ensuite, l'estimateur de densité SVM mono-classe sera détaillé. Enfin, la solution trouvée sera étudiée avant de clôturer cette section par l'analyse des capacités et des propriétés de cet estimateur.

2.3.2.1 Problématique de détermination de contour

Considérons $\chi_L = \{X_1, ..., X_L\}$ un ensemble de données distribuées suivant une loi de densité de probabilité P. L'objectif consiste à déterminer une fonction contour optimal renfermant toutes les données dans un domaine de volume minimal. La plupart des méthodes d'estimation de contour proposées dans la littérature se basent sur des modèles paramétriques trop rigides et conduisent à des approximations aberrantes ou erronées [Devroye & Wise,

1980; Muller, 1992; Walther 1997]. Pour formaliser la problématique d'estimation de contour, la notion de fonction quantile a été définie [Einmahl, & Mason, 1992.].

Définition 2.5 : Soit $\tilde{\Omega}$ une classe de sous ensembles mesurables sur l'ensemble $\chi_L = \{X_1,...,X_L\}$ des données distribuées suivant une loi P et v une fonction à valeurs réelles définie sur $\tilde{\Omega}$, V est une fonction quantile définie sur le triplet $(P,v,\tilde{\Omega})$ si :

$$V(\mu) = \inf\{v(\varpi) \mid P(\varpi) - \mu \geq 0,\ \varpi \in \tilde{\Omega}\},\quad \mu \in]0,1] \qquad 2.27$$

Intuitivement, $V(\mu)$ caractérise le plus petit sous ensemble contenant une masse de probabilité égale à μ. Il est clair que l'exploitation directe de ce formalisme n'est pas évidente.

En 1999, une technique plus pratique est proposée pour résoudre la problématique d'estimation de contour optimal [Tax & Duin, 1999]. Cette technique appelée SVD (*Support Vector Domain*) a été initialement développée pour la recherche de modèle de contour hyper-sphérique. La solution trouvée est ensuite étendue aux contours complexes en utilisant l'astuce de substitution de noyau. Supposons que le domaine décrit par les données X_i correspond à un modèle hyper-sphérique de centre A et de rayon R (Figure 2.2). La recherche de l'hyper-sphère de volume minimal contenant toutes les données est posée comme un problème d'optimisation exprimé par :

Minimiser $\quad R^2$

Sous contrainte $\quad \|X_i - A\|^2 \leq R^2,\quad \forall i \in [L]$

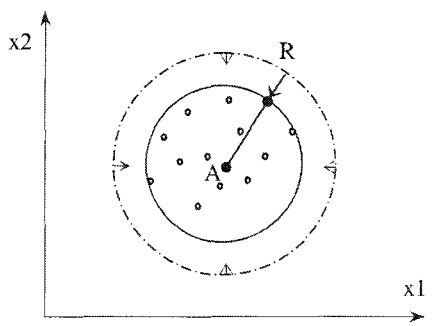

Figure 2.2 : Estimation de contour circulaire en 2D

Il convient de noter que le contour hyper-sphérique déterminé dans l'espace χ ne correspond pas au volume minimal du domaine décrit par les données (sur-apprentissage). En effet, dans la plupart des cas, la distribution des données caractérise des contours non sphériques. En utilisant l'astuce de substitution du noyau, l'idée du SVD a été exploitée avec succès pour développer l'estimateur de support de distribution dans l'espace RKHS.

2.3.2.2 Estimation de support de distribution dans l'espace RKHS

Développé pour la classification SVM mono-classe, l'estimateur de densité à l'aide de méthodes à noyau est présenté dans [Schölkopf & al. 2001]. Soit l'ensemble de données $\chi_L = \{X_1,...,X_L\}$ et $\kappa(\bullet,\bullet)$ un noyau défini positif dans l'espace RKHS $\Gamma = \phi(\chi)$, le noyau $\kappa(\bullet,\bullet)$ satisfait donc à la condition suivante :

$$\forall X_1, X_2 \in \chi,\quad \kappa(X_1,X_2) = \langle \phi(X_1), \phi(X_2) \rangle_\Gamma \geq 0 \qquad 2.28$$

On fait une hypothèse supplémentaire sur le noyau $\kappa(\cdot,\cdot)$ de telle sorte que :

$$\forall X \in \chi, \quad \kappa(X,X) = \langle \phi(X), \phi(X) \rangle_{\Gamma} = 1 \qquad 2.29$$

Le noyau gaussien satisfait aux deux conditions précédentes. En effet, on a :

$$0 \leq \kappa(X_1, X_2) = \exp\left(-\lambda \|X_1 - X_2\|\right) \leq 1, \quad \forall X_1, X_2 \in \chi \qquad 2.30$$

L'intérêt de l'utilisation d'un tel noyau est de permettre une interprétation géométrique simple et intuitive pour la recherche des solutions dans l'espace Γ. En effet, les conditions (2.28) et (2.29) imposent que toutes les données images $\phi(X)$ se situent sur un quart de cercle de rayon 1 dans l'espace Γ (Figure 2.3). Ce résultat est mis en évidence par les propriétés :

P1) $\langle \phi(X), \phi(X) \rangle_{\Gamma} = 1 \quad \Rightarrow \quad \|\phi(X)\|_{\Gamma} = 1, \quad \forall X \in \chi$ (i.e Rayon = 1)

P2) $\langle \phi(X_1), \phi(X_2) \rangle_{\Gamma} \geq 0 \quad \Rightarrow \quad \cos\left(\phi(X_1), \phi(X_2)\right) \in [0,1], \quad \forall X_1, X_2 \in \chi$

$$\Rightarrow \quad \text{angle}\left(\phi(X_1), \phi(X_2)\right) \in [0, \frac{\pi}{2}]$$

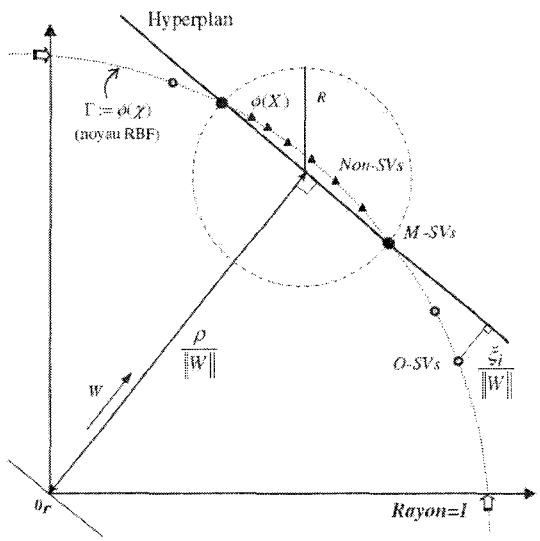

Figure 2.3: Représentation géométrique de la transformation ϕ en utilisant un noyau gaussien. RKHS : cercle de rayon 1. Trois groupes de données relativement à l'hyperplan de séparation : Non-SVs (internes au contour), M-SVs (sur la marge i.e sur l'hyperplan), O-SVs (avec erreurs).

Remarque : L'orthogonalité dans l'espace Γ avec l'utilisation du noyau gaussien, n'est obtenue que pour des vecteurs infiniment distants dans l'espace des données χ.

Rappelons que dans ce problème de classification mono-classe, l'objectif consiste à déterminer un contour de densité encapsulant toutes les données dans un domaine à volume minimal dans l'espace χ. Ceci se traduit dans l'espace Γ par l'estimation d'un hyperplan (support de distribution) très loin de l'origine 0_Γ et qui repousse toutes les données images dans un arc de cercle (Figure 2.3). L'équation de cet hyperplan est définie dans Γ par :

$$\langle W, \phi(X) \rangle_{\Gamma} - \rho = 0 \qquad 2.31$$

$W \in \Gamma$ est un vecteur normal à l'hyperplan et $\rho \in \Re_+$ l'offset de cette fonction.

Sachant que la marge entre l'origine 0_{Γ} et l'hyperplan de séparation est égale à $\rho / \|W\|_{\Gamma}$, l'estimation du contour optimal est posé comme un problème d'optimisation équivalent à :

$$\underset{W,\rho}{\text{Maximiser}} \quad \rho - \frac{1}{2}\langle W, W \rangle_{\Gamma} \qquad 2.32$$

$$\text{Sous contrainte} \quad \langle W, \phi(X) \rangle_{\Gamma} - \rho \geq 0 \qquad 2.33$$

La solution de ce problème d'optimisation force toutes les données à l'intérieur de la classe. Cette technique appelée version « marge dure » est très sensible aux données bruitées. Afin de permettre une estimation plus robuste du support de distribution, on utilise la version à « marge souple » [Chen & al., 2005]. Dans cette version, la contrainte (2.33) est relaxée en autorisant quelques données (bruit) hors du contour avec des erreurs de relâchement $\xi_i \geq 0$ (*slack errors*) de telle sorte que :

$$\xi_i > 0 \quad \text{si} \quad \langle W, \phi(X_i) \rangle_{\Gamma} - \rho < 0 \qquad 2.34$$

$$\xi_i = 0 \quad \text{si} \quad \langle W, \phi(X_i) \rangle_{\Gamma} - \rho \geq 0 \qquad 2.35$$

L'influence de la relaxation est pénalisée par rapport à la maximisation de la marge en la pondérant par une constante $C = 1/\nu.L$ dans le problème d'optimisation. $\nu \in [0,1]$ est un paramètre contrôlant la fraction des données autorisées en dehors du contour. De cette façon, le problème d'optimisation s'exprime par :

$$\underset{W,\rho,\xi_i}{\text{Minimiser}} \quad \underbrace{\frac{1}{2}\langle W, W \rangle_{\Gamma} - \rho}_{\text{Régularisation}} + \underbrace{\frac{1}{\nu L}\sum_{i=1}^{L} \xi_i}_{\text{Erreurs}} \qquad 2.36$$

$$\text{Sous contraintes} \quad \begin{cases} \langle W, \phi(X_i) \rangle_{\Gamma} - \rho \geq -\xi_i \\ \xi_i \geq 0 \end{cases} \quad \forall X_i \in \chi \qquad 2.37$$

2.3.2.3 Méthode de résolution

Le problème d'optimisation précédent est un critère quadratique à contraintes linéaires qui peut être traité en utilisant la technique classique du Lagrangien. A travers cette technique, le critère à minimiser s'obtient en introduisant les coefficients de Lagrange α_i, β_i (réels tous positifs) suivant la formulation suivante :

$$L(W, \rho, \xi, \alpha, \beta) = \frac{1}{2}\langle W, W \rangle_{\Gamma} + \frac{1}{\nu.L}\sum_{i=1}^{L} \xi_i - \rho - \sum_{i=1}^{L} \alpha_i . \left(\langle W, \phi(X_i) \rangle_{\Gamma} - \rho + \xi_i \right) - \sum_{i=1}^{L} \beta_i . \xi_i \qquad 2.38$$

Après dérivation par rapport aux variables primaires, on obtient :

$$\frac{\partial L(W,\rho,\xi,\alpha,\beta)}{\partial W} = W - \sum_{i=1}^{L} \alpha_i.\phi(X_i) = 0 \Rightarrow W = \sum_{i=1}^{L} \alpha_i.\phi(X_i) \quad 2.39$$

$$\frac{\partial L(W,\rho,\xi,\alpha,\beta)}{\partial \rho} = -1 + \sum_{i=1}^{L} \alpha_i = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i = 1 \quad 2.40$$

$$\frac{\partial L(W,\rho,\xi,\alpha,\beta)}{\partial \xi_i} = \frac{1}{\nu.L} - \alpha_i - \beta_i = 0 \Rightarrow \alpha_i = \frac{1}{\nu.L} - \beta_i \leq \frac{1}{\nu.L} \quad 2.41$$

Ensuite, à partir des conditions KKT[5], on a :

$$\alpha_i.\left(\langle W,\phi(X_i)\rangle_\Gamma - \rho + \xi_i\right) = 0 \quad 2.42$$

$$\beta_i.\xi_i = 0 \quad 2.43$$

En remplaçant W par sa valeur (2.39) dans la formule (2.38) et en prenant en compte les conditions KKT (2.42) et (2.43), on obtient la formulation duale du Lagrangien exprimée à partir des variables duales α_i (uniquement) :

$$\underset{\alpha}{\text{Minimiser}} \quad \frac{1}{2}\sum_{i}^{L}\sum_{j}^{L} \alpha_i.\alpha_j.\kappa(X_i,X_j) \quad 2.44$$

$$\text{Sous contraintes} \quad \begin{cases} \sum_{i=1}^{L} \alpha_i = 1 \\ 0 \leq \alpha_i \leq 1/\nu.L \end{cases} \quad 2.45$$

Ce critère est également quadratique et à contraintes linéaires. Il s'optimise donc numériquement [Kaufmann, 1999; Musicant, 2000]. Le choix de la méthode de résolution dépend du problème à traiter. Pour les applications en ligne, la technique SMO (*Sequentiel Minimal Optimisation*) est très adaptée pour sa rapidité d'exécution [Platt, 1999]. Les codes de quelques algorithmes d'optimisation se trouvent sur le site www.kernel-machines.org.

2.3.2.4 Étude de la solution

Après la résolution du problème d'optimisation (2.44), on obtient toutes les valeurs des α_i dont la plupart sont nulles. En remplaçant le paramètre W par son expression (2.39), la fonction d'apprentissage s'exprime simplement avec les vecteurs support SV_i qui sont les données correspondant aux α_i non nuls :

$$f(X) = \sum_i \alpha_i \kappa(X,SV_i) - \rho \quad 2.46$$

Par ailleurs, en utilisant l'équation de l'hyperplan, il est possible de déterminer la valeur de l'offset ρ. En effet, on a : $\{SV \in \chi / \langle W,\phi(SV)\rangle_\Gamma - \rho = 0\}$. Ainsi, l'offset ρ s'exprime par :

[5] KTT : Conditions de Karush Kuhn Tuker [Kuhn & al., 1950]

$$\rho = \sum_i \alpha_i \kappa(SV_c, SV_i) \qquad 2.47$$

SV_c étant un vecteur support choisi de façon quelconque parmi les SV_i.

Suivant les valeurs des α_i, les données X_i se distinguent en trois catégories selon leurs positions par rapport à l'hyperplan (figure 2.3). Ce résultat est mis en évidence à partir de l'étude du critère dual (2.44). Pour cela, ce critère est reformulé en y ajoutant la contrainte $\sum_{i=1}^{L} \alpha_i = 1$ avec ρ comme coefficient de Lagrange. On obtient ainsi :

$$\text{Minimiser} \quad G(\alpha, \rho) = \frac{1}{2} \sum_i^L \sum_j^L \alpha_i . \alpha_j . \kappa(X_i, X_j) - \rho \left(\sum_{i=1}^{L} \alpha_i - 1 \right) \qquad 2.48$$

$$\text{Sous contrainte} \quad 0 \le \alpha_i \le 1/\nu . L \qquad 2.49$$

L'expression du gradient de $G(\alpha, \rho)$ en α_i conduit au résultat suivant :

$$\frac{\partial G(\alpha, \rho)}{\partial \alpha_i} = \sum_{j=1}^{L} \alpha_j . \kappa(X_i, X_j) - \rho = f(X_i) \qquad 2.50$$

Il est montré dans [Cauwenberghs & Paggio, 2001; Desobry, 2004] que l'étude du critère dual du Lagrangien en fonction des poids α_i permet d'établir les résultats suivants :

$$\frac{\partial G(\alpha, \rho)}{\partial \alpha_i} = f(X_i) = \begin{cases} > 0 \rightarrow \alpha_i = 0 & \rightarrow X_i \in E_{SV}^{non} \quad (\xi_i = 0) \\ = 0 \rightarrow 0 < \alpha_i < 1/\nu . L & \rightarrow X_i \in E_{SV}^{M} \quad (\xi_i = 0) \\ < 0 \rightarrow \alpha_i = 1/\nu . L & \rightarrow X_i \in E_{SV}^{O} \quad (\xi_i > 0) \end{cases} \qquad 2.51$$

avec :

- E_{SV}^{non} : l'ensemble de données classées sans erreurs *non-SV* (i.e internes au contour)
- E_{SV}^{M} : l'ensemble de données de la marge *M-SV* (i.e sur contour)
- E_{SV}^{O} : l'ensemble de données classées avec erreurs *O-SV* (i.e. externes au contour)

La fonction de décision ne dépend donc que des données des ensembles E_{SV}^{M} et E_{SV}^{O} ; ces données sont les vecteurs support notés SV_i. Cette propriété est intéressante car elle réduit considérablement la complexité de calcul. Ainsi, la fonction de décision s'écrit simplement :

$$\text{sgn}\left(f(X)\right) = \text{sgn}\left(\sum_i \alpha_i \kappa(X, SV_i) - \sum_i \alpha_i \kappa(SV_c, SV_i) \right) \qquad 2.52$$

L'estimation de support de distribution avec l'approche noyau est illustrée sur la figure 2.4.

Après l'étude de l'estimateur de densité à l'aide des méthodes à noyau, la section suivante a pour but de montrer sa flexibilité et ses capacités à décrire une large variété de classifieurs suivant le réglage du coefficient ν et du paramètre λ du noyau gaussien.

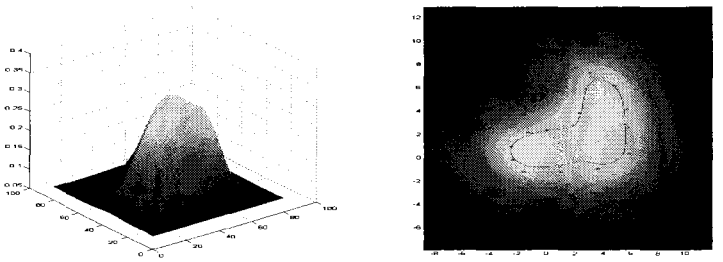

Figure 2.4 : Estimation de densité à l'aide des méthodes à noyau (200 données bruitées ; $\nu = 0,2$ Noyau gaussien de paramètre $\delta = 1$).

2.3.2.5 Relaxation et propriétés du noyau gaussien

Tout d'abord, on commence par étudier l'influence du coefficient ν contrôlant l'effet de la relaxation, c'est-à-dire, l'impact des variables de relâchement. Ensuite, l'influence du réglage du paramètre λ du noyau gaussien sera analysée.

a) ν-propriété

Théorème 2.1 : En supposant que $\nu \neq 0$, ces propositions sont vraies :

(i) $\nu \geq \frac{card(E_{SV}^{O})}{L}$: ν est la borne supérieure de la fraction des outliers.

(ii) $\nu \leq \frac{card(E_{SV}^{M})}{L}$: ν est la borne inférieure de la fraction des vecteurs de support

(iii) Si en plus, les données sont indépendamment générées suivant une loi de probabilité continue, alors ν est asymptotiquement égal à la fraction des vecteurs supports et à celle des outliers.

Ce théorème est prouvé dans [Schölpkopf & al., 2000].

1) Si $\nu \to 0$, $C \to \infty$. En analysant le critère (2.36) et (2.37), on remarque que la pénalisation d'erreurs devient infinie. On retrouve le critère de maximisation à « marge dure » (Figure 2.5.a) :

- En supposant qu'aucune contrainte n'est imposée sur l'offset ρ, ce dernier peut donc devenir un nombre négatif infiniment grand ($\rho << 0$), mais le problème reste tout de même faisable. Tous les points seront acceptés à l'intérieur du contour.
- Par contre, si la contrainte $\rho \geq 0$ est imposée, le terme de régularisation devient donc insignifiant devant le risque empirique. La minimisation du critère (2.36) force toutes les erreurs ξ_i à 0. Ce qui se traduit par l'absence d'outliers. Toutes les données seront donc à l'intérieur du contour. Quant aux α_i, ils pourraient diverger car leur borne supérieure devient infinie dans l'équation (2.45).

2) Si par contre on fixe $\nu = 1$, le classifieur SVM mono-classe se comporte comme un estimateur de Parzen (Figure 2.5.b). En effet, à partir du théorème 2.1, on déduit que toutes

les données deviennent des outliers, les contraintes sont saturées avec : $\alpha_i = \frac{1}{L}$, $\forall i \in [L]$. f s'exprime sur une fenêtre de Parzen :

$$f(X) = \sum_{i=1}^{L} \alpha_i . \kappa(X, X_i) = \frac{1}{L} \sum_{i=1}^{L} \kappa(X, X_i) \qquad 2.53$$

La figure 2.5 illustre les cas de classification à « marge dure » et l'estimation de densité de Parzen obtenus avec le classifieur One-Class-SVM en utilisant deux réglages différents du paramètre ν. Il ressort de cette étude que le paramètre ν offre une grande flexibilité à l'estimateur de densité par les méthodes à noyau. Un choix judicieux de ν permet d'optimiser le résultat de l'estimation dans un environnement bruité. Cependant, la qualité de ce résultat n'est garantie que si le paramètre λ du noyau gaussien est convenablement choisi.

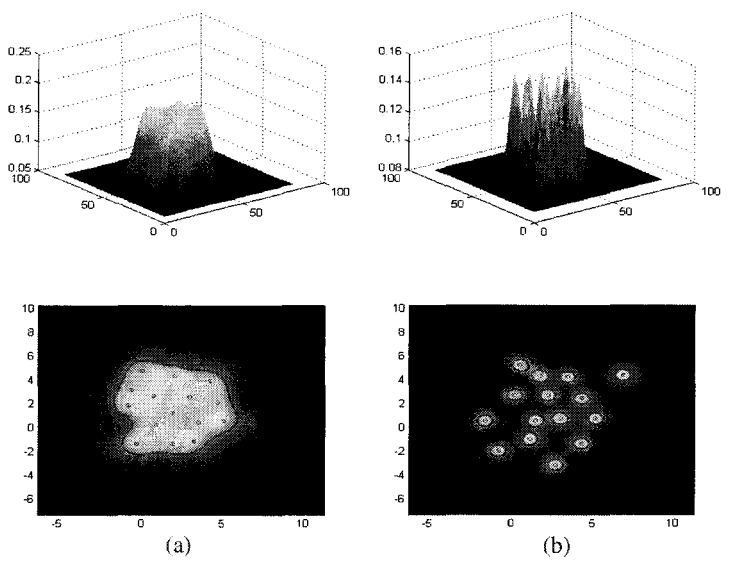

Figure 2.5 : Influence de la relaxation : (a) si $\nu = 0$, estimateur de densité équivalent à celui de la version marge dure (pas d'outliers). (b) si $\nu = 1$, estimateur équivalent à celui de Parzen (chaque donnée est un outlier). Noyau gaussien de paramètre $\lambda = 0,5$; simulation avec 15 données bruitées.

b) Propriété du noyau gaussien

Les performances élevées du noyau gaussien dans le cadre de la classification sont prouvées par diverses études [Schölkopf & al., 1997; Tu, 2003]. Ce noyau dispose en effet, de quelques propriétés intéressantes qui motivent son choix dans la plupart des problèmes.

1) **Interprétation géométrique** de l'espace RKHS : Avec un noyau gaussien, la transformation $\phi : \chi \to \Gamma$ construit l'espace RKHS en projetant toutes les données sur un quadrant de cercle de rayon 1. Cette interprétation géométrique est très intéressante car elle facilite le raisonnement pour la mise en œuvre des méthodes d'apprentissage [Desobry & al., 2005] et [Amadou & al., 2005a].

2) Toutes les données images sont **linéairement indépendantes** dans l'espace RKHS :
Théorème 2.2 : La matrice de Gram du noyau gaussien est de rang plein [Micchelli, 1986]. Ce théorème traduit le fait que tous les points images $\phi(X_i)$ sont linéairement indépendants dans l'espace RKHS, s'il n'existe pas deux données X_i identiques dans χ. Les données $X_i \in \chi$ sont transformées dans un espace RKHS de dimension infinie sans être confondues.

3) **Universalité et Flexibilité** : Toute fonction continue peut être approximée (en norme infinie) par la combinaison des fonctions $\kappa(.,X)$, $X \in \chi$ de l'espace de Hilbert. De même, avec suffisamment de données, il est possible de construire n'importe quelle fonction analytique. Par ailleurs, en fonction du choix du paramètre du noyau gaussien, on peut construire divers classifieurs :

- $\lambda \to 0$, tous les points se retrouvent dans un domaine de volume infiniment petit dans l'espace de Hilbert. Le noyau se comporte ainsi comme un *classifieur linéaire* dans χ.
- $\lambda \to \infty$, tous les points sont orthogonaux dans l'espace de Hilbert, chaque point est donc dissimilaire des autres. Le noyau se comporte comme un *classifieur kpp-voisin* avec $k = 1$.
- En variant λ, le noyau permet de tester plusieurs types de classifieurs par d'innombrables combinaisons intermédiaires.

2.3.3 Discussions sur les SVM et méthodes à noyau

Les techniques SVM sont développées pour la classification supervisée. Elles sont fondées sur un principe de classification à marge maximale, généralement suivant une approche de séparation. Les SVM utilisent des fonctions d'apprentissage à noyau ayant de bonnes performances de modélisation. L'estimation de fonctions complexes est facilitée par la transformation des données dans un espace de Hilbert doté du produit scalaire et rendant possible l'utilisation de calculs de l'algèbre linéaire. Le modèle de classification optimal est obtenu en minimisant le risque de Vapnik qui prend en compte à la fois les erreurs et le choix du modèle adéquat parmi une classe de fonctions grâce à la technique de régularisation. Utilisés d'abord pour la classification binaire, les algorithmes SVM ont ensuite été proposés pour les problèmes multi-classes. La plupart de ces algorithmes utilisent des hyperplans de séparation pour représenter les frontières des classes. Des solutions optimales sont obtenues à travers la résolution d'un problème d'optimisation. Mais ces solutions restent intéressantes dans un cadre de classification basé sur l'approche de séparation.

L'estimateur de densité à l'aide des méthodes à noyau offre un modèle intéressant pour représenter une classe suivant l'approche par modélisation. Ce modèle non-paramétrique est donné par une fonction d'apprentissage à noyau qui ne fait aucun *a priori* sur la forme de classes. Cependant, l'estimateur de densité à l'aide des méthodes à noyau est réservé à la classification mono-classe et se base sur un critère d'optimisation qui nécessite que toutes les données soient disponibles dans une base complète. Il existe quelques techniques qui généralisent la méthode du One-Class-SVM aux problèmes multi-classes [Ben-Hur & al.,

2002; Tu, 2003; Sun & Huang, 2003]. Mais, les algorithmes SVM développés jusqu'à présent, sont loin de résoudre les problèmes de classification dynamique. En effet, très peu d'algorithmes SVM disposent de procédures de modélisation adaptative.

2.4 Conclusion

Nous venons d'étudier les modèles de mélange et les méthodes à noyau dans un contexte de classification de données spatiales. Notre intérêt est particulièrement porté sur leurs capacités à modéliser des classes plus ou moins complexes. L'étude des méthodes de classification nous a permis d'analyser leurs principes et d'établir leurs limitations pour la classification de données non-stationnaires.

Le modèle de mélange apporte un formalisme intéressant pour construire un modèle de classification de données spatiales notamment en environnement multi-classe. En effet, ces modèles décrivent la distribution de données en utilisant une loi de densité de probabilité caractérisée par autant de composantes qu'il y a de classes dans la partition. La loi de densité gaussienne est le modèle de distribution le plus répandu. Il permet d'introduire une approximation satisfaisante dans beaucoup d'applications. En plus, les éléments mathématiques du modèle gaussien sont relativement bien maîtrisés. L'estimation des paramètres d'un modèle de mélange se fait par la maximisation du critère de vraisemblance. Malgré que ce critère soit bien établi pour caractériser l'homogénéité des classes, il ne garantit pas un meilleur résultat si la loi de densité choisie (*a priori*) ne convient pas à la distribution des données. Or, cette distribution étant inconnue, les méthodes de maximisation de vraisemblance ne sont pas à l'abri des problèmes d'optima locaux. Par ailleurs, la plupart des algorithmes proposés dans les modèles de mélange se limitent à la classification non-supervisée. Ils nécessitent la connaissance *a priori* de toutes les données d'apprentissage et se trouvent être relativement gourmands en temps de calcul.

Quant aux méthodes à noyau, elles ont des fondements théoriques solides et des bonnes capacités de généralisation. Les fonctions d'apprentissage à noyau se présentent comme des modèles non-paramètriques capables de modéliser des classes complexes. Elles sont déterminées dans un espace de Hilbert à noyau reproduisant (RKHS) dont les propriétés sont très intéressantes pour la classification. En effet, grâce à l'astuce de substitution de noyau, il est possible de traiter les problèmes de non-linéarités qui posent énormément de difficultés dans beaucoup d'applications. La transformation des données dans un espace RKHS doté du produit scalaire, rend possible l'utilisation des calculs de l'algèbre linéaire et la géométrie analytique. La détermination de formes complexes peut donc se faire avec un temps de calcul raisonnable. Les techniques SVM, très utilisées en classification pour l'estimation des fonctions d'apprentissage à noyau, se basent sur la minimisation du risque de Vapnik. Selon la théorie de Vapnik, un bon apprentissage passe non seulement, par la minimisation des erreurs (Risque empirique), mais aussi par le choix des modèles les plus simples et adaptés au détriment de ceux trop complexes (Régularisation). Cette approche est différente de celle des méthodes de modèles de mélange dont l'objectif consiste à maximiser la vraisemblance en

fixant *a priori* un modèle de densité (partie 2.2). Cependant, la plupart des algorithmes SVM sont généralement conçus pour la classification supervisée.

On a montré dans ce chapitre que les modèles de mélange et les fonctions d'apprentissage à noyau sont des outils de modélisation très performants en classification. Cependant, du point de vue de la problématique de données non-stationnaires, la plupart des algorithmes proposés (de type EM ou SVM) pour l'estimation de modèles de classification se trouvent très limités. En effet, ils ne disposent pas de procédures de modélisation adaptative. Le chapitre suivant est consacré à l'étude de méthodes récursives permettant l'adaptation de modèles par l'incorporation continue de nouvelles données. Quelques algorithmes incrémentaux développés pour la mise à jour de modèles de mélanges et de fonctions d'apprentissage à noyau seront présentés.

Chap. 3 : Techniques de modélisation adaptative

3.1 Introduction

L'étude menée dans le chapitre précédent a permis de mettre en évidence les capacités de modélisation de deux méthodes statistiques dans le cadre de la classification de données spatiales. Les modèles de mélange offrent un formalisme théorique intéressant basé sur une décomposition de la densité de données sous forme de plusieurs distributions homogènes. Les méthodes à noyau quant à elles, utilisent des fonctions d'apprentissage à noyau qui sont des modèles non-paramétriques capables de modéliser des formes complexes. Malgré les potentialités de ces deux méthodes, très peu de techniques sont proposées pour permettre la mise à jour de leurs modèles dans le contexte du traitement des données non-stationnaires. Si ces données sont disponibles (connaissances *a priori*), elles sont traitées **hors ligne** suivant un ordre chronologique. Mais dans beaucoup de problèmes réels, la base d'apprentissage est incomplète, les données sont acquises **en ligne** de façon continue et chaque donnée doit être traitée dès qu'elle se présente. Du point de vue de la classification dynamique, les données non-stationnaires sont des données datées caractérisées par une structure de représentation temporelle. Le **temps** est un facteur véritablement essentiel en modélisation adaptative, notamment pour la prise en compte des changements en environnement dynamique [Fabiani, 1996 ; Fleury, 1998]. La représentation temporelle offre deux avantages considérables :

- L'introduction d'un modèle d'évolution naturelle des données et des connaissances dans l'espace de représentation.
- La possibilité de déterminer des résultats intermédiaires dont l'interprétation est généralement utile pour la prise de décision et/ou la prédiction dans divers problèmes.

L'apprentissage du modèle de classification à partir de données non-stationnaires se fait au travers de l'incorporation continue de nouvelles informations au fur et à mesure qu'elles deviennent disponibles. La difficulté réside dans la mise en œuvre de règles récursives permettant l'adaptation du modèle de classification face aux modifications locales et aux évolutions avec glissements. Rappelons que dans la problématique de la classification dynamique, la procédure d'adaptation récursive de classes présente des risques d'optima

locaux pouvant affecter considérablement la qualité du modèle de classification (chapitre 1, section 1.2). Cette adaptation se fait suivant un processus d'apprentissage itératif qui utilise à chaque pas les modèles de classes déjà formées et les nouvelles données pour mettre à jour de la partition dynamique. Il faut par ailleurs, noter que le déroulement de ce processus se fait avec des ressources limitées (complexité algorithme, capacité mémoire,...). Dans la littérature, il existe plusieurs techniques développées pour l'adaptation récursive de modèles [Mendel & Fu, 1970; Kasabov, 2003]. La plupart de ces techniques sont proposées dans le contexte de l'apprentissage actif et de l'apprentissage incrémental. L'apprentissage actif est largement utilisé dans les problèmes de fouilles de données [Tong, 2001]. Ce type d'apprentissage nécessite au préalable la mise en œuvre d'une procédure sélective visant à choisir dans une base complète, les données les plus informatives pour adapter le modèle. L'apprentissage incrémental quant à lui, vise à incorporer chaque nouvelle donnée dès qu'elle se présente [Murray, 1995; Liu & al., 2000]. La différence fondamentale entre ces deux types d'apprentissage réside dans la considération effectuée sur les données (sélection ou non) en amont de la procédure de mise à jour du modèle [Vijayakumar, 1998]. L'apprentissage incrémental est donc mieux adapté aux problèmes en ligne (base incomplète). Dans le cadre de la classification dynamique, chaque nouvelle donnée agit de manière plus ou moins forte sur la partition et contribue donc à la redéfinition du modèle de classification. Ces contributions se traduisent essentiellement par l'apport de nouvelles connaissances, la modification des connaissances existantes ou encore leurs remises en cause.

Ce chapitre dédié aux méthodes de modélisation adaptative pour la classification, est divisé en trois parties principales :

Les deux premières parties présentent des méthodes récursives. La première partie est consacrée aux méthodes itératives d'approximation stochastique. Après l'étude théorique de ces méthodes basées sur le gradient, on expose quelques algorithmes incrémentaux basés sur le gradient et permettant la mise à jour de modèles de mélange gaussien et des fonctions d'apprentissage à noyau. Dans la seconde partie, on étudie les méthodes récursives exactes avant d'illustrer leurs mises en œuvre pour l'estimation itérative du modèle gaussien et la classification incrémentale du SVM mono-classe. L'étude des méthodes récursives de la première et de la seconde partie a pour intérêt d'apporter les outils nécessaires servant à la mise en œuvre de la procédure 2 de la description générique (chapitre 1, partie 1.4).

Enfin, la dernière partie étudie quelques techniques de fusion et de scission développées en classification. Cette partie présente le competitive-EM qui, bien qu'il ne soit pas un algorithme récursif, est doté de règles de fusion et de scission. Ces règles interviennent dans les procédures 3 et 4 de la description générique (chapitre 1, partie 1.4) et participent donc à la modélisation adaptative de la partition dynamique.

3.2 Méthodes itératives d'approximation stochastique

L'estimation itérative du modèle de la partition dynamique est un problème d'apprentissage adaptatif. On peut préciser que :

- La notion d'*apprentissage* est dédiée à la tâche qui consiste à déterminer un modèle de connaissances à partir des informations contenues dans toutes les données disponibles.
- La notion d'*apprentissage adaptatif* fait référence au processus de mise à jour d'un modèle de connaissances par l'incorporation séquentielle de nouvelles informations.

L'apprentissage adaptatif est généralement posé comme un problème d'optimisation. En effet dans la plupart des situations, l'objectif est de concevoir un système qui permet d'estimer un modèle en optimisant un critère défini sur l'ensemble des données. Il faut donc trouver des méthodes itératives qui assurent la convergence vers un optimum. Parmi les méthodes exposées dans la littérature, nous proposons d'analyser brièvement les plus utilisées.

La méthode du gradient [Bottou, 1998] est la plus utilisée. Elle repose sur le calcul du gradient du critère à optimiser. C'est une méthode simple et très rapide dont la convergence est démontrée. Cependant, lorsque l'on s'approche de l'optimum, la vitesse de convergence diminue très fortement car le gradient du critère tend vers 0.

La méthode de Newton [Minoux, 1983] utilise le gradient et l'inverse de la matrice hessienne (matrice des dérivées secondes) pour minimiser un critère. C'est une méthode itérative très efficace au voisinage du minimum. Mais la convergence de la méthode de Newton n'est garantie que si la matrice hessienne est définie positive. De plus, le calcul de la matrice hessienne à chaque itération, démande un temps d'exécution important.

La méthode de Quasi-Newton [Minoux, 1983] est proposée pour pallier les inconvénients de la méthode de Newton. L'idée consiste à générer une séquence de matrices symétriques définies positives pour approximer l'inverse de la matrice hessienne. De cette façon, la méthode converge vers un minimum du critère. Mais, l'approximation de l'inverse de la matrice hessienne à chaque itération reste une tâche complexe à réaliser et couteuse en temps de calcul.

La méthode de Leverberg-Marquardt [Bishop, 1995] effectue un compromis entre la méthode du gradient et la méthode de Newton. Dans sa mise en œuvre, elle fait intervenir astucieusement le gradient et la matrice hessienne de façon à faire converger le critère suivant la direction du gradient et/ou celle de la méthode de Newton. Cependant, cette méthode n'est fiable que s'il y a très peu de paramètres d'initialisation. Comme les méthodes précédentes, elle reste couteuse en temps de calcul.

Parmi ces méthodes, nous portons plus d'intérêt à la méthode du gradient pour sa rapidité et sa simplicité de mise en œuvre. En plus de ces avantages, les résultats qu'elle obtient sont satisfaisants pour la plupart des problèmes. Les autres méthodes convergent avec plus de précision, mais cette précision est obtenue au détriment de la simplicité de mise en œuvre et de la vitesse d'exécution.

Nous détaillons dans la suite de cette partie la méthode du gradient stochastique et sa mise en œuvre. Après cette étude, on présente dans la section 3.2.2, les versions incrémentales des algorithmes *k-mean* et CEM puis le NORMA (*Naïve Online Rreg. Minimisation Algorithm*) basés sur le gradient stochastique.

3.2.1 Approximation stochastique par la méthode du gradient

Introduit dans les années 50 [Robbins & Monro, 1951], la méthode du gradient stochastique a permis d'élaborer des systèmes adaptatifs dans divers domaines de l'apprentissage artificiel [Amari, 1993; Balakrishmann, 2003; Coulon, 2002]. Un algorithme basé sur le gradient stochastique produit des résultats successifs en optimisation un critère d'erreurs (risque d'apprentissage) ou en maximisant un critère de qualité (critère de vraisemblance). Considérons un problème d'apprentissage dans lequel l'on souhaite estimer le modèle F en optimisant un critère global défini *a posteriori* sur l'ensemble des données d'apprentissage, sous la forme générale suivante :

$$Z[F] = \tilde{E}_{\chi}[J(F,X)] = \int_{\chi} J(F,X)dP(X) \qquad 3.1$$

où $J(F,X)$ est une mesure d'erreur ou de qualité estimée localement sur la donnée $X \in \chi$, et $\tilde{E}_{\chi}[\bullet]$ l'espérance mathématique associée à la variable aléatoire $X \in \chi$. Le critère global Z est ainsi défini par l'intégrale sur toutes les données distribuées suivant une loi de densité P inconnue. La technique du **gradient batch** est une première solution proposée pour optimiser le critère global Z [Bottou, 1998]. Bien que cette technique produit des estimations successives du modèle F, elle ne convient pas aux problèmes en ligne. En effet, elle utilise comme terme d'adaptation le gradient du critère Z estimé sur l'ensemble de données supposées connues *a priori*. Selon la nature du critère à optimiser, on distingue deux mises en œuvre en ligne du gradient stochastique : la **descente du gradient** et **la montée du gradient**.

3.2.1.1 La descente du gradient

Si J est une fonction coût mesurant les erreurs d'apprentissage, l'objectif consiste donc à estimer le modèle F en minimisant le critère global (3.1). On peut remarquer que dans cette situation, le critère Z correspond à un risque d'apprentissage. Le gradient recherche la minimisation du critère en utilisant comme terme d'adaptation le gradient de la fonction coût J. Si l'on souhaite minimiser un risque, par exemple, cette fonction correspond à une mesure d'erreur ξ évaluée localement sur chaque donnée X_t à l'instant t. Ainsi, l'adaptation du modèle F par la technique de la descente du gradient se fait en utilisant la formule itérative :

$$F^{t+1} = F^t - \eta \frac{\partial}{\partial F} \xi(F^t, X_{t+1}) \qquad 3.2$$

$\eta > 0$: ratio d'apprentissage.

La figure 3.1 illustre la procédure d'estimation récursive du modèle en utilisant la technique de la descente du gradient.

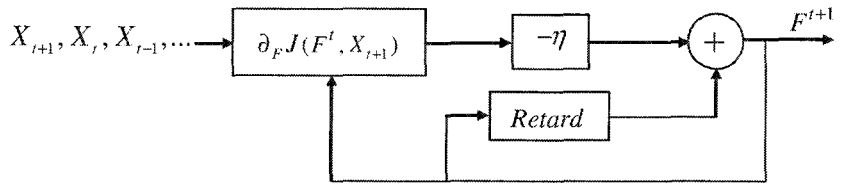

Figure 3.1 : Procédure de la descente du gradient en ligne.

3.2.1.2 La montée du gradient

Dans certains problèmes d'apprentissage, l'objectif consiste à maximiser un critère de type vraisemblance. C'est le cas par exemple pour la classification avec les modèles de mélange (chapitre 2, section 2.2.2). Dans ces situations, la fonction J du critère (3.2) correspond donc à une mesure de qualité q évaluée localement sur chaque donnée X_t à l'instant t. La technique de la montée de gradient produit des estimations successives du modèle F en maximisant le critère Z. Cette maximisation se fait en utilisant la formule itérative suivante :

$$F^{t+1} = F^t + \eta \frac{\partial}{\partial F} q(F^t, X_{t+1}) \qquad 3.3$$

Hormis les signes, la formule de la descente du gradient et celle de la montée du gradient ont la même expression. Intuitivement, le signe (-) dans la formule (3.2) diminue l'influence des erreurs dans l'estimation du modèle, tandis que le signe (+) dans la formule (3.3) vise à augmenter la qualité du modèle. A partir de la figure 3.1, on peut illustrer la procédure de la montée du gradient en changeant la contre-réaction $-\eta$ par une réaction positive $+\eta$.

La méthode du gradient stochastique en ligne à travers la mise en œuvre de la descente du gradient ou celle de la montée du gradient ne nécessite pas la connaissance *a priori* de la loi densité P. Cependant, elle suppose que la fonction J soit différentiable pour pouvoir calculer le terme d'adaptation.

3.2.1.3 Gradient Stochastique Généralisé

Dans les situations où la fonction J du critère (3.1) n'est pas différentiable partout sur χ, la technique du gradient stochastique généralisé est une solution intéressante. Cette technique est une simple généralisation qui consiste à optimiser le critère global Z en choisissant une fonction H dont l'espérance mathématique satisfait à la condition suivante :

$$\tilde{E}_{\chi}[H(F,X)] = \frac{\partial Z[F]}{\partial F} \qquad 3.4$$

Selon l'objectif du problème d'apprentissage, les règles de mise à jour du gradient stochastique généralisé s'expriment en utilisant comme terme d'adaptation $H(F^t, X_{t+1})$:

- Minimisation du critère Z : $F^{t+1} = F^t - \eta H(F^t, X_{t+1})$
- Maximisation du critère Z : $F^{t+1} = F^t + \eta H(F^t, X_{t+1})$

Cette formulation générale étend le champ d'application du gradient stochastique en ligne à divers problèmes [Bottou, 2004]. En utilisant une fonction H quelconque satisfaisant la condition (3.4), on s'affranchit de la dérivation de la vraie fonction coût du critère global (3.1). Cette technique a été utilisée pour la mise en œuvre de l'algorithme CEM incrémental exposé dans la section 3.2.2.2. Sous certaines conditions de régularité et de différentiabilité des fonctions Z, H, Bottou et Bengio [1995] montrent que le gradient stochastique généralisé converge. La convergence vers un optimum global du critère est possible, mais n'est pas toujours garantie car elle dépend d'une manière forte de l'initialisation (figure 3.2).

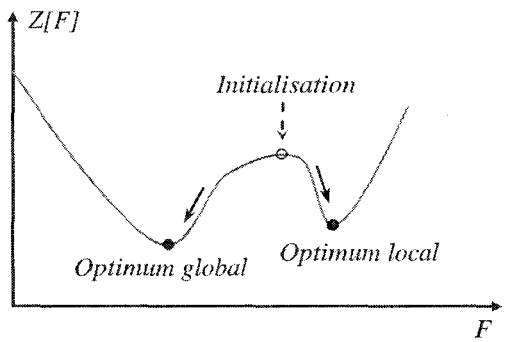

Figure 3.2 : Algorithme du Gradient Stochastique : la convergence vers un optimum local ou global dépend de l'initialisation de l'algorithme.

3.2.1.4 Choix du ratio d'apprentissage

Le choix du ratio d'apprentissage η est prépondérant mais délicat à réaliser. Si ce paramètre est choisi trop petit, très peu d'importance sera accordée aux informations récentes. Ainsi donc, l'adaptation du modèle devient très lente et moins réactive aux variations brusques. En revanche, si le ratio η est choisi trop grand, le risque de non-convergence devient trop élevé. Le ratio d'apprentissage η peut être choisi constant ou variable en fonction du temps. Lorsque l'on souhaite une convergence vers un modèle optimal fixe, on utilise un ratio décroissant vers la valeur 0. Dans les situations de suivi d'évolutions, un ratio d'apprentissage η constant est mieux adapté. Le choix de η est souvent effectué dans l'intervalle $]0,1[$ après plusieurs essais et corrections [Coulon, 2002].

Le gradient stochastique est une méthode itérative convergente [Bottou & Bengio, 1995 ; Bottou, 2004]. Contrairement à la méthode du gradient batch, le gradient stochastique en ligne échappe plus facilement aux optima locaux en environnement bruité. En effet, la sensibilité au bruit produit un comportement d'instabilité bénéfique à la convergence [Genevieve & Todd, 1993]. Ce comportement est très atténué dans l'apprentissage avec le gradient batch à cause du cumul des gradients calculés à chaque instant sur toutes les données.

D'autres qualités de la méthode du gradient stochastique en ligne sont sa simplicité de mise en œuvre et sa rapidité. Par ailleurs, grâce à un oubli progressif des informations anciennes, cette méthode permet le suivi des évolutions du modèle. Le gradient stochastique offre un modèle approximatif qui reste tout de même satisfaisant dans la plupart des applications.

3.2.2 Algorithmes incrémentaux basés sur le gradient stochastique

Dans cette section, nous allons étudier quelques algorithmes incrémentaux basés sur la méthode du gradient stochastique. Ces algorithmes développés pour la classification de données spatio-temporelles, sont conçus pour l'apprentissage récursif de classes au travers de la mise à jour itérative de leurs modèles. On peut résumer la démarche suivie pour développer ces algorithmes en deux étapes essentielles. La première étape consiste à formaliser le problème de classification sous forme d'un problème d'optimisation à travers la définition d'un critère global. Ce critère est défini en utilisant une fonction coût mesurant l'erreur ou la qualité de classification. La seconde étape consiste à déterminer les règles de mise à jour itératives en se basant sur la méthode du gradient qui recherche la minimisation ou la maximisation du critère global selon les objectifs du problème. Il s'agit précisément de calculer le terme d'adaptation des formules itératives du gradient. Cette démarche est illustrée à travers l'étude des versions incrémentales du *k-means*, du CEM et l'étude du NORMA.

3.2.2.1 L'algorithme k-means incrémental

L'algorithme du *k-means* [McQueen, 1967] est très utilisé en classification automatique pour la détermination de partition optimale. Cet algorithme effectue la classification de données en optimisant le critère global (2.6) (chapitre 2 section 2.2.1). Cet algorithme également appelé de centres mobiles, est développé avec une procédure d'apprentissage séquentiel qui se formule simplement en utilisant la technique du gradient stochastique. Considérons la partition Π caractérisée par un modèle de classification $\Im$ de paramètre $\Theta = \{\bar{X}_1, ..., \bar{X}_m, ..., \bar{X}_M\}$, où $\bar{X}_m$ est le centre de la classe C_m, le principe du *k-means* consiste à déterminer la partition de données en minimisant le critère suivant :

$$Z[\Im] = \tilde{E}_X\left[\frac{1}{2}\left(X - \bar{X}_m\right)^T\left(X - \bar{X}_m\right)\right] \qquad 3.5$$

Dans le cadre de la classification séquentielle à partir de données datées X_t, le critère (3.5) se définit à chaque instant t à partir d'une fonction coût J (3.1) dont le gradient est :

$$\partial_\Theta\left[J(\Theta_m^t, X_{t+1})\right] = -\left(X_{t+1} - \bar{X}_m^t\right) \qquad 3.6$$

La règle de mise à jour itérative du *k-means* est établie à partir de la descente du gradient (3.2) et conduit à l'adaptation séquentielle des centres mobiles $\bar{X}_m$ à partir de la formule suivante :

$$\bar{X}_m^{t+1} = \bar{X}_m^t + \eta.\left(X_{t+1} - \bar{X}_m^t\right) \qquad 3.7$$

avec $\eta = \frac{1}{n_m + 1}$ où n_m est la cardinalité de C_m à l'instant t dans le cas de partition dure.

La convergence de cet algorithme est démontrée dans [Bottou & Bengio, 1995]. L'algorithme *k-means* définit une structure de partition avec des classes de forme sphérique qui est un cas particulier de modèle mélange gaussien.

3.2.2.2 L'algorithme CEM incrémental

Des travaux récents ont permis de développer quelques algorithmes incrémentaux pour l'estimation itérative des paramètres d'un modèle de mélange afin de pallier à certaines limitations des méthodes de maximisation de la vraisemblance [Neal & Hinton, 1998; Same, 2004]. Il s'agit principalement des versions séquentielles des algorithmes EM et CEM (chapitre 2, section 2.3). L'EM étant un algorithme de classification floue, nous proposons d'étudier l'algorithme CEM incrémental (ou incrémentiel) dans le cadre de la classification à l'aide de modèles de mélange gaussien. La mise en œuvre séquentielle du CEM est effectuée en utilisant la méthode du gradient stochastique. Rappelons que l'algorithme du CEM estime les paramètres $\{\Theta_m\}_{m=1,\ldots,M}$ d'un modèle de classification $\Im$ en maximisant la log-vraisemblance classifiante définie de la manière suivante :

$$\log L_C[\Im] = \sum_{m=1}^{M} \sum_{X \in C_m} \log\left(\pi_m \varphi(X, \Theta_m)\right) \qquad 3.8$$

On peut mettre cette expression sous la forme du critère maximisé par le gradient stochastique (équation 3.4) de telle sorte qu'on obtienne :

$$Z[\Im] = \tilde{E}_\chi\left[\max_{1 \le m \le M} \log\left(\pi_m \varphi(X, \Theta_m)\right)\right] \qquad 3.9$$

De cette façon, on identifie facilement la fonction J (3.1) qui est ici une mesure de qualité et dont le gradient donne le terme d'adaptation utilisé dans la formule de la montée du gradient :

$$\frac{\partial}{\partial \Theta}\left[J(\Theta, X)\right] = \frac{\partial}{\partial \Theta} \max_{1 \le m \le M} \log\left(\pi_m \varphi(X, \Theta_m)\right) \qquad 3.10$$

L'estimation itérative de paramètres Θ_m se fait ainsi à chaque instant t de telle sorte que :

$$\Theta_m^{t+1} = \Theta_m^t + \eta \frac{\partial}{\partial \Theta} \max_{1 \le m \le M} \log\left(\pi_m \varphi\left(X_{t+1}, \Theta_m^t\right)\right) \qquad 3.11$$

Malheureusement, l'utilisation de cette formule pose des problèmes de différentiabilité. En effet, le gradient n'existe pas pour certaines valeurs à cause de la fonction max qui n'est pas partout continue et différentiable. Pour pallier ce problème, on utilise une fonction H vérifiant les conditions de régularité définies par Bottou [1998] de telle sorte qu'on puisse

appliquer la technique du gradient stochastique généralisé pour maximiser le critère de vraisemblance classifiante (3.8). Ce terme d'adaptation $H(\Phi, X)$ est défini tel que :

$$H(\Theta, X) = \begin{cases} \partial_\Theta J(\Theta, X) & \text{si } \partial_\Theta J(\Theta, X) \text{ existe} \\ 0 & \text{sinon} \end{cases} \qquad 3.12$$

Dans la pratique, forcer H à la valeur zéro lorsque le gradient de J n'est pas calculable pour une donnée signifie que l'on passe à la donnée suivante.

Rappelons que dans le cas de mélange gaussien chaque paramètre $\Theta_m = (\bar{X}_m, \Sigma_m)$, où $\bar{X}_m$ et Σ_m sont respectivement la moyenne et la matrice de covariance de la classe C_m. Et, chaque composante est pondérée par sa proportion π_m dans le mélange. Deux conditions de probabilité sont imposées sur les proportions : $0 < \pi_m < 1$ et $\pi_1 + ... + \pi_M = 1$. Ces conditions ne sont pas toujours satisfaites au cours de la mise à jour des proportions du mélange gaussien par la méthode du gradient. Afin de garantir la stabilité numérique, la solution proposée consiste à reparamétrer ces proportions en utilisant la transformation $\tilde{\pi}_m = \log(\pi_m / \pi_M)$. Ainsi, les formules récursives du CEM incrémental pour la mise à jour des paramètres du mélange gaussien sont les suivantes [Samé & al., 2004] :

$$w_m^{t+1} = w_m^t + \eta_t \left(z_m^{t+1} - \pi_m^t \right), \quad m < M \qquad 3.13$$

$$\pi_m^{t+1} = \frac{\exp(w_m^{t+1})}{1 + \sum_{m=1}^{M-1} \exp(w_m^{t+1})}, \quad m < M \text{ et } \pi_M^{t+1} = \frac{1}{1 + \sum_{m=1}^{M-1} \exp(w_m^{t+1})} \qquad 3.14$$

$$\bar{X}_m^{t+1} = \bar{X}_m^t + z_m^{t+1} \eta_t \left[\Sigma_m^{t\,-1} \left(X_{t+1} - \bar{X}_m^t \right) \right] \qquad 3.15$$

$$\Sigma_m^{t+1} = \Sigma_m^t + z_m^{t+1} \eta_t \left[\frac{1}{2} \Sigma_m^{t\,-1} \left(\left(X_{t+1} - \bar{X}_m^t \right) \left(X_{t+1} - \bar{X}_m^t \right)^T \Sigma_m^{t\,-1} - I_D \right) \right] \qquad 3.16$$

$$\text{avec} \begin{cases} z_m^{t+1} = 1, & \text{si } m = \arg\max_{1 \le m \le M} \log\left(\pi_m \varphi\left(X_{t+1}, \Phi_m^t \right) \right) \\ z_m^{t+1} = 0, & \text{sinon} \end{cases}$$

Le CEM incrémental a été utilisé dans une application réelle dans le cadre d'un projet d'émission acoustique du CETIM (CEntre Technique des Industries Mécanique) [Samé, 2004]. Dans cette application, l'algorithme est utilisé hors ligne pour l'apprentissage de classes statiques en incorporant de façon séquentielle les données disponibles dans une base connue *a priori*. L'algorithme CEM incrémental est plus performant que la version classique en terme de temps de calcul. Cependant, l'utilisation de cet algorithme nécessite la connaissance *a priori* du nombre de classes.

Dans l'Annexe A.1, on montre que les formules récursives de mise à jour de la plupart des modèles de mélange paramétriques peuvent être déterminées de façon générale en appliquant directement la méthode du gradient stochastique sur la forme exponentielle. En

effet, cette forme décrit une famille regroupant le mélange gaussien et bien d'autres lois de densités (exponentielle, poisson, dirichlet, gamma,...) [Mora, 1986; Letac & Mora, 1990].

3.2.2.3 NORMA : Classification séquentielle mono-classe.

Dans le chapitre 2, section 2.3.2, nous avions étudié l'estimateur de densité à l'aide des méthodes à noyau. A l'issue de cette étude, les performances de cet estimateur non-paramétrique ont été montrées. L'algorithme One-Class-SVM proposé pour la classification mono-classe dispose de bonnes bases théoriques, mais n'est pas adapté aux problèmes d'apprentissage incrémental. Dans la présente section, nous allons présenter un algorithme itératif appelé NORMA (*Naïve On-line $R_{reg.}$[1] Minimisation Algorithm*) pour la mise à jour récursive du support de distribution dans l'espace RKHS [Kivinen & al., 2004].

Tout d'abord, nous allons présenter la formulation générale de l'algorithme NORMA. Ensuite, nous exposons son application à la classification séquentielle mono-classe en montrant ses capacités à estimer de façon itérative les paramètres de la fonction d'apprentissage à noyau qui caractérise la distribution de données.

a) Algorithme NORMA

Rappelons que dans les méthodes à noyau, l'objectif consiste à estimer une fonction d'apprentissage à noyau F_ω en minimisant le critère de risque régularisé (2.23) (Chapitre 2, section 2.3.1.1). Considérant l'ensemble de données $\{X_i\}_{i=1,\dots,L}$, ce risque s'exprime par :

$$R_{reg}[F_\omega, \chi_L] = \frac{1}{L}\sum_{i=1}^{L} \xi(F_\omega, X_i) + G[F_\omega] \qquad 3.17$$

L'utilisation de ce critère suppose donc la connaissance *a priori* de toutes les données X_i. Pour les problèmes en ligne, on construit un critère coût instantané à partir du risque régularisé (3.17) de façon à mesurer à chaque instant t l'erreur d'apprentissage de F_ω^t localement sur la donnée X_t. Ceci est rendu possible en remplaçant le terme de risque empirique (erreur moyenne) par l'erreur instantanée $\xi(F_\omega^t, X_t)$ et en utilisant la norme de F_ω^t comme terme de régularisation dans le critère (3.17). De cette façon, on définit le Risque Instantané R_{inst} exprimé comme suit :

$$R_{inst}[F_\varpi^t, X_{t+1}] = \xi(F_\varpi^t, X_{t+1}) + \frac{a}{2}\left\|F_\varpi^t\right\|_\Gamma^2 \qquad 3.18$$

$a > 0$ est une constante de pénalisation.

L'algorithme NORMA est développé en utilisant la technique de la descente du gradient pour la minimisation du risque instantané (3.18). Ainsi, la mise à jour itérative de la fonction F_ω^t est effectuée à partir de la formule suivante :

[1] $R_{reg.}$: Regularised Risk (Risque régularisé)

$$F_{\omega}^{t+1} = F_{\omega}^{t} - \eta \frac{\partial}{\partial F_{\omega}} R_{inst}\left[F_{\omega}^{t}, X_{t+1}\right] \qquad 3.19$$

Après la dérivation du risque instantané (3.18), on obtient la relation :

$$F_{\omega}^{t+1} = (1-\eta a)F_{\omega}^{t} - \eta\kappa(X_{t+1},.)\left[\partial_{F_{\omega}}\xi(F_{\omega}^{t}, X_{t+1})\right] \qquad 3.20$$

D'ores et déjà, on peut constater que le terme $(1-\eta a)$ ne doit pas être négatif sinon l'algorithme diverge. Par ailleurs, sachant que $\eta > 0$ et $a > 0$, une condition nécessaire pour que l'algorithme fonctionne correctement est : $0 < \eta < a^{-1}$.

En notant F la fonction caractérisant le terme d'expansion du noyau, les formules itératives du NORMA sont établies à partir de la formule générale (3.20) de telle sorte que la mise jour de paramètres de la fonction F se fasse de la façon suivante :

$$F^{t}(X) = \sum_{i=1}^{t-1} \alpha_i \kappa(X, SV_i) \rightarrow \begin{cases} \alpha_i^{t+1} = (1-\eta a)\alpha_i^{t}, & i < t \\ \alpha_i^{t+1} = \eta\left(\frac{\partial}{\partial F}\xi(F^{t}, X_{t+1})\right), & i = t \end{cases} \qquad 3.21$$

Il est à noter que l'algorithme NORMA est initialisé avec l'hypothèse $F^{0} = 0$.

L'algorithme NORMA est appliqué à divers problèmes d'apprentissage en ligne : classification binaire, estimation de densité et régression [Kivinen & al. 2004]. Il suffit pour cela de connaître la fonction d'apprentissage et de définir la fonction coût ξ adéquate.

b) Application à la classification séquentielle mono-classe

La classification SVM mono-classe a été introduite à l'origine pour l'estimation de densité à l'aide de méthodes à noyau [Schölkopf & al. 2001]. Elle vise à déterminer une fonction contour encapsulant l'ensemble de données X_i dans un domaine de volume minimal (chapitre 2, section 2.3.2). Ce contour est défini par une fonction d'apprentissage à noyau de la forme suivante :

$$f(X) = F(X) - \rho = \sum_{i} \alpha_i \kappa(X, SV_i) - \rho \qquad 3.22$$

avec les coefficients $\alpha_i \in \Re^{+}$, l'offset $\rho > 0$ et les vecteurs support SV_i.

En partant des formules (3.20) et (3.21), on peut déterminer les règles de mise à jour des paramètres de la fonction f pour les problèmes d'estimation séquentielle de densité. La fonction coût de *Hinge* s'est avérée la mieux adaptée pour pénaliser les erreurs d'un tel estimateur de densité [kivinen & al., 2004]. Cette fonction coût est définie par :

$$\xi(f, X) = \max(0, \rho - F(X)) - \nu\rho \qquad 3.23$$

avec $\nu \in [0,1]$ le coefficient fixant la fraction de vecteurs support (version SVM *soft-margin*)

La fonction coût ξ est continue et dérivable par morceaux. Ses dérivées par rapport à F sont :

$$\frac{\partial}{\partial F}\xi(f,X)=\begin{cases}-1 & \text{si } f(X)<0\\ 0 & \text{si } f(X)\geq 0\end{cases} \qquad 3.24$$

Par ailleurs, l'offset ρ est une fonction temporelle dont l'adaptation est aussi effectuée en utilisant la formule du gradient (3.19). Les dérivées de la fonction coût par rapport à ρ sont :

$$\frac{\partial}{\partial \rho}\xi(f,X)=\begin{cases}1-\nu & \text{si } f(X)<0\\ -\nu & \text{si } f(X)\geq 0\end{cases} \qquad 3.25$$

Finalement, des équations (3.21), (3.24), (3.25), il en découle les formules itératives de mis à jour des paramètres α_i et ρ de la fonction f pour l'estimation séquentielle de densité :

$$\begin{cases}\begin{cases}\alpha_i^{t+1}=(1-\eta)\alpha_i^t, & i<t\\ \alpha_{i=t}^{t+1}=\eta \text{ [resp. 0]} & \text{si } f^t(X_{t+1})<0 \text{ [resp. } f^t(X_{t+1})\geq 0]\end{cases}\\ \begin{cases}\rho^{t+1}=\rho^t+\eta(1-\nu) & \text{si } f^t(X_{t+1})<0\\ \rho^{t+1}=\rho^t-\eta\nu & \text{si } f^t(X_{t+1})\geq 0\end{cases}\end{cases} \qquad 3.26$$

Ces formules sont établies en fixant le paramètre de pénalité de l'équation (3.21) : $a=1$.

Lorsque la quantité de données est trop grande, on limitera le nombre de calculs en effectuant l'apprentissage sur une fenêtre de taille τ. Cela permet également d'oublier les informations obsolètes. Les paramètres α_i décroissent ainsi de façon exponentielle au cours du temps entraînant l'élimination des vecteurs support de faibles poids pour ne conserver que ceux d'indices satisfaisant la relation $\max(1,t-\tau)\leq i\leq t$. Cette technique de mise à jour itérative est simple et très rapide. Cependant, des inconvénients liés à un accroissement monotone de l'offset ρ causent une augmentation anormale des erreurs cumulées lors de l'apprentissage des données non-stationnaires. Ceci provoque un phénomène indésirable dans un contexte d'apprentissage des classes évolutives et limite l'utilisation du NORMA à un apprentissage itératif dédié aux cibles plus ou moins statiques.

3.3 Méthodes récursives de mise à jour exacte

Dans la partie précédente les méthodes récursives basées sur la méthode du gradient stochastique ont été présentées. Ces méthodes produisent des approximations successives du modèle de classification en optimisant un critère global. Outre ces méthodes basées sur le gradient stochastique, il existe d'autres méthodes récursives de mise à jour de modèles développées pour les problèmes d'apprentissage séquentiel [Merwe & Wan, 2001; Pronzato & Pazman, 2004]. Dans cette section, nous nous intéressons aux méthodes récursives qui calculent de façon itérative le modèle exact. Ces méthodes de mise à jour exacte que nous appelons simplement **méthodes récursives exactes** sont assez souvent utilisées en apprentissage incrémental mais rarement présentées dans la littérature sous un formalisme théorique. En effet, compte tenu de leurs diversités il est difficile de proposer une démarche générale décrivant leur principe de mise en œuvre. Nous allons néanmoins tenter de donner une formulation générale des méthodes récursives exactes avant de présenter quelques algorithmes incrémentaux basés sur ces méthodes.

3.3.1 Formulation générale

Dans les problèmes d'apprentissage séquentiel, on cherche à produire des estimations successives du modèle de connaissances en utilisant les connaissances déjà acquises et les informations des nouvelles données. Les méthodes récursives exactes sont généralement élaborées à partir d'un calcul analytique ayant pour but de déterminer la récurrence sur le modèle. Afin de préciser cette problématique, on considère les hypothèses suivantes :

- Soit un ensemble de données $\chi_L = \{X_1, ..., X_L\}$ indépendamment distribuées suivant une loi de densité de probabilité P. On souhaite calculer un modèle F de paramètre $\Theta \in \Omega$ à partir des données de χ_L.
- Supposons qu'il existe un estimateur théorique $\hat{E}(\cdot|P)$ défini de χ vers Ω et capable d'estimer le paramètre Θ du modèle F à partir des données X_i supposées connues *a priori* de telle sorte que : $\Theta = \hat{E}(\{X_1, ..., X_L\}|P)$.

L'objectif consiste à déterminer la récurrence sur le modèle de façon à écrire l'estimateur $\hat{E}(\cdot|P)$ sous une forme récursive utilisable dans les situations d'adaptation de modèles. Nous distinguons ces situations en deux types :

1. Ajout d'information : Soit $\Theta^t = \hat{E}(\{X_1, ..., X_t\}|P)$ le paramètre estimé à partir des données datées $\{X_1, ..., X_{t+1}\}$, on suppose qu'il existe une fonction A et deux coefficients r et β telle que la mise à jour de Θ s'effectue à l'aide d'une formule récursive de la forme :

$$\Theta^{t+1} = r.\Theta^{t} + \beta.A\left(\Theta^{t}, X_{t+1}\right) \qquad 3.27$$

La fonction A est dite opérateur d'ajout d'information. r et β sont des coefficients scalaires ou matriciels. La formule récursive (3.27) convient spécifiquement à l'adaptation de modèles à travers l'incorporation de nouvelles données.

2. Ajout et Retrait d'information : Considérant les paramètres $\Theta^{N} = \tilde{\mathrm{E}}(\{X_{t-N+1},...,X_{t}\}|P)$ et $\Theta^{N*} = \tilde{\mathrm{E}}(\{X_{t-N+2},...,X_{t+1}\}|P)$ estimés sur une fenêtre de données datées de taille N, on suppose qu'il existe une fonction AR et deux coefficients r' et β' de telle sorte qu'on puisse écrire :

$$\Theta^{N*} = r'.\Theta^{N} + \beta'.AR\left(\Theta^{N}, X_{t-N+1}, X_{t+1}\right) \qquad 3.28$$

La fonction AR est dite opérateur d'ajout et de retrait d'information sur une fenêtre glissante. r' et β' sont des coefficients scalaires ou matriciels. La formule récursive (3.28) est particulièrement adaptée à la mise à jour de modèles dans les situations d'évolutions avec glissements par l'incorporation de nouvelles données et l'oubli des données anciennes.

La détermination de la récurrence sur le modèle pour la mise en œuvre des formules récursives (3.27) et (3.28) se fait souvent au travers de calculs mathématiques rigoureux et difficiles. En effet, il s'agit de calculer les opérateurs A et AR et les coefficients r, β, r', et β' à partir de la décomposition analytique de l'estimateur théorique $\tilde{\mathrm{E}}(\cdot|P)$. Afin de distinguer les méthodes itératives produisant une approximation de modèle, nous introduisons la définition des méthodes récursives exactes.

> **Définition 3.2 :** Une méthode récursive est dite exacte si la mise à jour du paramètre Θ estimé sur toutes les données connues jusqu'à l'instant t en utilisant des formules récursives de la forme (3.27) et (3.28) produit un résultat identique à celui calculé par l'estimateur théorique $\tilde{\mathrm{E}}(\cdot|P)$ sur les mêmes données.

De cette définition, on déduit qu'une méthode récursive exacte est convergente. En effet, le modèle estimé par une telle méthode et celui donnée par l'estimateur théorique étant identiques, l'erreur de convergence entre ces deux modèles est donc nulle. Dans les formules (3.27) et (3.28), les cœfficients r, β, r', et β' dépendent généralement du nombre de données constituant le modèle. La plupart des méthodes basées sur le gradient stochastique ne produisent pas une mise à jour exacte du modèle mais plutôt une approximation réglable par le choix du ratio d'apprentissage η.

Avant de terminer cette section, noter qu'il n'existe pas de « recette » pour la mise en œuvre des méthodes récursives exactes. Néanmoins, une astuce très utilisée consiste à revenir à la définition de l'estimateur théorique $\tilde{\mathrm{E}}(\cdot|P)$ afin de déterminer la récurrence sur le modèle. Cette détermination est effectuée en décomposant analytiquement son expression de façon à extraire les opérateurs A et AR et les coefficients r, β, r', et β' (équation 3.28). La mise en œuvre de certaines méthodes récursives exactes est exposée dans la section suivante.

3.3.2 Algorithmes incrémentaux basés sur les méthodes récursives exactes

Les méthodes récursives exactes étudiées dans cette section, sont développées pour la conception des algorithmes incrémentaux basés sur les modèles de mélanges gaussien et les fonctions d'apprentissage à noyau. Il s'agit tout d'abord d'exposer la méthode récursive de mise à jour exacte des paramètres du modèle gaussien. Ensuite, l'estimateur séquentiel de densité à l'aide de méthodes à noyau ou plus précisément la version incrémentale de l'algorithme SVM mono-classe sera étudiée.

3.3.2.1 Estimation récursive du modèle gaussien

A travers l'étude du CEM incrémental (section 3.2.2.2), nous avons montré comment est effectuée l'adaptation des classes gaussiennes à l'aide de la méthode du gradient stochastique. Cette méthode produit de façon itérative un résultat approximatif du modèle. Il existe une méthode récursive de mise à jour exacte développée par Vasseur & al. [1988] qui a permis l'estimation itérative des paramètres d'une classe gaussienne. Elle a récemment été exploitée avec succès dans le cadre de la classification dynamique [Lecoeuche & Lurette, 2003; 2004]. Nous proposons d'étudier cette méthode récursive à travers la détermination de la récurrence sur le modèle gaussien. Considérons le modèle $\varphi(\bullet,\Theta)$ représentant une classe gaussienne notée C, le paramètre $\Theta=(\bar{X},\Sigma)$, avec $\bar{X}$ la moyenne et Σ la matrice de covariance de la classe C. Soit $\{X_1,...X_L\}$ l'ensemble des données de la classe C, on suppose que ces données sont distribuées suivant une loi normale P, l'estimateur théorique de densité gaussienne $\hat{\mathrm{E}}(\bullet|P)$ calcule le paramètre $\Theta=(\bar{X},\Sigma)$ de la façon suivante :

$$\Theta=\hat{\mathrm{E}}(X_1,...,X_L|P) \rightarrow \begin{cases} \bar{X}=\frac{1}{L}\sum_{i=1}^{L}X_i \\ \Sigma=\frac{1}{L-1}\sum_{i=1}^{L}(X_i-\bar{X})(X_i-\bar{X})^T \end{cases} \qquad 3.29$$

A partir de ces formules nous allons examiner comment se fait la mise à jour exacte du modèle gaussien à travers d'une part l'incorporation séquentielle des nouvelles données et d'autre part l'ajout et le retrait d'informations sur une fenêtre glissante.

a) Ajout d'information :

Considérons un ensemble de données datées $\{X_1,...,X_t\}$ à partir duquel est défini le modèle $\Theta^t=\hat{\mathrm{E}}(X_1,...,X_t|P)$. On souhaite calculer le modèle Θ^{t+1} à partir du modèle Θ^t et de la donnée X_{t+1}. L'objectif consiste donc à déterminer la récurrence sur la moyenne et la matrice de covariance de façon à écrire l'estimateur $\hat{\mathrm{E}}(\bullet|P)$ sous la forme récursive (3.27).

- Détermination de la récurrence sur la moyenne : $\bar{X}^t \rightarrow \bar{X}^{t+1}$

A partir de la définition de l'estimateur $\hat{\mathrm{E}}(\cdot|P)$ dans l'expression (3.29), on peut écrire :

$$\bar{X}^{t+1} = \frac{1}{t+1}\sum_{i=1}^{t+1} X_i = \frac{1}{t+1}\left(\sum_{i=1}^{t} X_i + X_t + 1\right) = \frac{1}{t+1}\left(t.\bar{X}^t + X_{t+1}\right) \qquad 3.30$$

Ce qui se traduit simplement par la formule récurrente suivante :

$$\bar{X}^{t+1} = \bar{X}^t + \frac{1}{t+1}\left(X_{t+1} - \bar{X}^t\right) \qquad 3.31$$

De la formule (3.27), on identifie $A\left(\Theta^t, X_{t+1}\right) = X_{t+1} - \bar{X}^t$ et les coefficients $r = 1$, $\beta = \frac{1}{t+1}$.

Remarque : La formule récursive (3.31) est celle utilisée dans l'algorithme *k-means* incrémental (section 3.2.2.1) pour l'adaptation des centres de classes (avec le ratio $\eta = \beta$). Il est donc possible dans certaines situations d'établir une méthode récursive de mise à jour exacte en utilisant le gradient stochastique avec un ratio d'apprentissage déterminé par le calcul et non choisi de façon empirique.

- Détermination de la récurrence sur la matrice de covariance : $\Sigma^t \rightarrow \Sigma^{t+1}$

De la même façon que pour la moyenne, on peut déterminer la récurrence sur la matrice de covariance afin d'effectuer la mise à jour itérative de Σ par l'ajout de nouvelles informations. En revenant à la définition (3.29) de l'estimateur gaussien $\hat{\mathrm{E}}(\cdot|P)$, on a :

$$\begin{aligned}\Sigma^{t+1} &= \frac{1}{t}\sum_{i=1}^{t+1}\left(X_i - \bar{X}^{t+1}\right)\left(X_i - \bar{X}^{t+1}\right)^T \\ &= \frac{1}{t}\left[\underbrace{\sum_{i=1}^{t}\left(X_i - \bar{X}^{t+1}\right)\left(X_i - \bar{X}^{t+1}\right)^T}_{A} + \underbrace{\left(X_{t+1} - \bar{X}^{t+1}\right)\left(X_{t+1} - \bar{X}^{t+1}\right)^T}_{B}\right]\end{aligned} \qquad 3.32$$

En remplaçant dans A la moyenne $\bar{X}^{t+1}$ par son expression (3.31), on obtient :

$$\begin{aligned}A &= \sum_{i=1}^{t}\left(\left(X_i - \bar{X}^t\right) - \frac{1}{t+1}\left(X_{t+1} - \bar{X}^t\right)\right)\left(\left(X_i - \bar{X}^t\right) - \frac{1}{t+1}\left(X_{t+1} - \bar{X}^t\right)\right)^T \\ &= \sum_{i=1}^{t}\left(X_i - \bar{X}^t\right)\left(X_i - \bar{X}^t\right)^T - 2\frac{1}{t+1}\left(\sum_{i=1}^{t}\left(X_i - \bar{X}^t\right)\right)\left(X_{t+1} - \bar{X}^t\right)^T + \frac{t}{(t+1)^2}\left(X_{t+1} - \bar{X}^t\right)\left(X_{t+1} - \bar{X}^t\right)^T\end{aligned}$$

On remarque que $\bar{X}^t$ est le centre de gravité des données X_i : $\sum_{i=1}^{t}\left(X_i - \bar{X}^t\right) = 0$. Ainsi :

$$A = (t-1)\Sigma^t + \frac{t}{(t+1)^2}\left(X_{t+1} - \bar{X}^t\right)\left(X_{t+1} - \bar{X}^t\right)^T \qquad 3.33$$

De même, en remplaçant dans B la moyenne $\bar{X}^{t+1}$ par son expression (3.31), on obtient :

$$B=\left(X_{t+1}-\bar{X}^{t}-\frac{1}{t+1}\left(X_{t+1}-\bar{X}^{t}\right)\right)\left(X_{t+1}-\bar{X}^{t}-\frac{1}{t+1}\left(X_{t+1}-\bar{X}^{t}\right)\right)^{T} \qquad 3.34$$

$$B=\left(\frac{t}{t+1}\right)^{2}\left(X_{t+1}-\bar{X}^{t}\right)\left(X_{t+1}-\bar{X}^{t}\right)^{T} \qquad 3.35$$

Enfin, des équations (3.32) (3.33) et (3.35) et après simplification, on obtient la formule récursive de mise à jour exacte de la matrice de covariance :

$$\Sigma^{t+1}=\frac{t-1}{t}\Sigma^{t}+\frac{1}{t+1}\left(X_{t+1}-\bar{X}^{t}\right)\left(X_{t+1}-\bar{X}^{t}\right)^{T} \qquad 3.36$$

En comparant (3.27) et (3.36), on a : $A\left(\Theta^{t},X_{t+1}\right)=\left(X_{t+1}-\bar{X}^{t}\right)\left(X_{t+1}-\bar{X}^{t}\right)^{T}$, $r=\frac{t-1}{t}$, $\beta=\frac{1}{t+1}$.

Les formules (3.31) et (3.36) de mise à jour de la moyenne et de la matrice de covariance permettent la prise en compte des modifications locales du modèle gaussien.

b) Ajout et retrait d'information :

Considérant un ensemble de données $\{X_{t-N+1},...,X_{t}\}$ à partir duquel est calculé Θ^{N} du modèle gaussien, l'objectif consiste cette fois-ci, à déterminer de façon récursive le paramètre $\Theta^{N^{*}}=\hat{\mathrm{E}}\left(X_{t-N+2},...,X_{t+1}|P\right)$ sur une fenêtre glissante de taille N en ajoutant la donnée X_{t+1} et en retirant la donnée X_{t-N+1} de l'ensemble d'apprentissage. Les calculs effectués pour la détermination de la récurrence sur le modèle sont présentés en Annexe A.3. Ici, nous ne donnons que les résultats de ces calculs que sont les formules récursives permettant la mise à jour itérative de la moyenne et de la matrice de covariance sur une fenêtre glissante.

- Détermination de la récurrence sur la moyenne : $\bar{X}^{N}\rightarrow\bar{X}^{N^{*}}$

La formule récursive de mise à jour de la moyenne est établie telle que :

$$\bar{X}^{N^{*}}=\bar{X}^{N}+\frac{1}{N}\left(X_{t+1}-X_{t-N+1}\right) \qquad 3.37$$

A partir de la formule (3.28), on identifie l'opérateur d'ajout et de retrait d'information $AR\left(\Theta^{N},X_{t-N+1},X_{t+1}\right)=X_{t+1}-X_{t-N+1}$ et les paramètres $r'=1$ et $\beta'=\frac{1}{N}$

- Détermination de la récurrence sur la matrice de covariance : $\Sigma^{N}\rightarrow\Sigma^{N^{*}}$

La mise à jour de la matrice de covariance est effectuée à partir de la formule récursive :

$$\Sigma^{N^{*}}=\Sigma^{N}+\Delta X B^{-1}\Delta X^{T} \qquad 3.38$$

Avec : $B=\frac{1}{N}\begin{pmatrix}1 & \frac{1}{N-1}\\ \frac{1}{N-1} & -\frac{N+1}{N-1}\end{pmatrix}$, $\Delta X=\left[X^{t+1}+\bar{X}^{N},\ X^{t-N+1}-\bar{X}^{N}\right]$

De la formule (3.28), on déduit que : $\beta'.AR\left(\Theta^N, X_{t-N+1}, X_{t+1}\right) = \Delta X B^{-1} \Delta X^T$ et $r' = 1$.

Les formules récursives (3.37) et (3.38) permettent le suivi de classes évolutives au travers de la mise à jour exacte de la moyenne et de la matrice de covariance. Ces formules de mise à jour du modèle gaussien sont utilisées avec succès pour le développement d'un algorithme de classification dynamique [Lurette, 2003; Lecoeuche & Lurette, 2003; 2004]. Une nouvelle version de cet algorithme sera présentée dans le chapitre 5.

3.3.2.2 SVM mono-classe incrémental

Dans la section 3.2.2.3, l'algorithme NORMA a été exposé pour la classification incrémentale mono-classe. Les formules itératives établies à partir du gradient stochastique dans l'espace RKHS permettent une approximation itérative de la fonction d'apprentissage à noyau pour l'estimation de densité. Les premiers travaux sur les SVM incrémentaux ayant permis le développement des méthodes récursives exactes ont été présentés récemment dans [Cauwenberghs & Poggio, 2001; Desobry, 2004]. Dans la problématique d'estimation de densité à l'aide de méthodes à noyau (chapitre 2, section 2.3.2), la fonction d'apprentissage à noyau est calculée en optimisant un critère de Lagrangien (équations 2.44, 2.45). L'idée introduite dans les techniques SVM incrémentales consiste à déterminer la récurrence sur la fonction d'apprentissage à noyau en respectant les contraintes de ce critère. En 2003, Gretton & Desobry [2003] propose une version séquentielle de l'algorithme One-Class-SVM qui est capable de calculer de façon récursive le support de distribution dans l'espace RKHS. Rappelons que ce support de distribution représente le contour de la classe dans l'espace de données χ. En supposant $\{X_1, \ldots, X_L\}$ les données distribuées suivant une loi de densité de probabilité inconnue P, le contour de la classe est défini par une fonction d'apprentissage à noyau (non-paramétrique). Cette fonction est déterminée par les vecteurs supports (i.e. les données X_i ayant des poids α_i non nuls) de telle sorte que (chapitre 2, section 2.3.2.4) :

$$f(\cdot, SV_i) = \sum_{j=1}^{\tau} \alpha_j . \kappa\left(\cdot, SV_j\right) - \rho \qquad 3.39$$

τ est le nombre de vecteurs support. Si l'on connaît les données *a priori*, les paramètres α_i et ρ sont déterminés par un estimateur de densité $\hat{E}(\cdot, P)$ qui est ici le classifieur One-Class-SVM. Cette estimation est effectuée en minimisant un critère d'optimisation (Lagrangien) équivalent au critère suivant :

$$D(\alpha, \rho) = \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j \kappa(X_i, X_j) + \rho \left(1 - \sum_{i=1}^{L} \alpha_i\right) \qquad 3.40$$

Sous contrainte : $0 \le \alpha_i \le \frac{1}{\nu . L}$

Selon cette formulation, α_i et ρ sont les coefficients de Lagrange. A l'optimum du critère (3.40), les paramètres α_i et ρ sont soumis à des conditions d'équilibre dit adiabatique[2]. Cet équilibre est exprimé en dérivant $D(\alpha,\rho)$ par rapport aux variables α_i et ρ de telle sorte que :

$$\frac{\partial D(\alpha,\rho)}{\partial \alpha_i} = \sum_{j=1}^{L} \alpha_j K(X_i, X_j) - \rho = 0 \qquad 3.41$$

$$\frac{\partial D(\alpha,\rho)}{\partial \rho} = 1 - \sum_{i=1}^{L} \alpha_j = 0 \qquad 3.42$$

Cauwenberghs & Poggio [2001] pose la problématique d'estimation récursive de la fonction f comme celle de la mise à jour exacte des paramètres α_i et ρ en respectant l'équilibre adiabatique traduit par les conditions (3.41) et (3.42). Cela signifie que lorsqu'on bouge (incrément ou décrément) une donnée X_c, tous les paramètres α_i , ρ doivent s'ajuster de façon à annuler les dérivées $\partial_\alpha D(\alpha,\rho)$ et $\partial_\rho D(\alpha,\rho)$. En supposant $\Delta\alpha_c$ la variation portée sur α_c (poids de X_c), tous les autres paramètres subiront des variations relatives pour rétablir l'équilibre (adiabatique) manière à avoir :

$$\begin{cases} \Delta\alpha_j = \beta_{\alpha_j} \Delta\alpha_c \\ \Delta\rho = -\beta_\rho \Delta\alpha_c \end{cases} \qquad 3.43$$

β_{α_j} et β_ρ sont des coefficients réels.

En posant $K_c = \left[X_c \bullet SV_j\right]_{j=1,\ldots,\tau}$ le vecteur de produits scalaires de X_c avec tous les vecteurs support SV_j et $K_{ij} = \left[SV_i \bullet SV_j\right]_{i,j=1,\ldots,\tau}$ la matrice de Gram, Gretton & Desobry [2003] montrent que les coefficients β_{α_j} et β_ρ se calculent de la manière suivante :

$$\begin{bmatrix} \beta_{\alpha_1} & \cdots & \beta_{\alpha_\tau} & \beta_\rho \end{bmatrix}^T = -\begin{bmatrix} 0 & 1_\tau^T \\ 1_\tau & \left[K_{ij}\right] \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ K_c \end{bmatrix} \qquad 3.44$$

Pour plus d'informations sur la mise en œuvre de l'algorithme One-Class-SVM incrémental, nous invitons le lecteur à consulter [Gretton & Desobry, 2003; Desobry, 2004]. Cet algorithme a été utilisé dans une application de détection de rupture dans les signaux audio [Desobry & al. 2005].

Contrairement à l'algorithme NORMA, l'algorithme One-Class-SVM incrémental est doté de règles récursives permettant la mise à jour exacte de la fonction d'apprentissage à noyau dans les problèmes de classification mono-classe. Cependant, les inconvénients majeurs de cet algorithme sont sa difficulté de mise en œuvre et sa complexité trop élevée pour envisager son utilisation dans les problèmes temps réel. En effet, la détermination des

[2] Terme adiabatique : analogie avec une loi de transformation réversible en Thermodynamique.

coefficients d'adaptation (3.44) introduit un coût de calcul important car elle nécessite à chaque instant l'inversion d'une matrice carrée de taille $\tau+1$.

L'étude menée dans les deux parties précédentes a permis d'appréhender deux catégories de méthodes récursives pour la mise à jour de modèles. Il s'agit d'une part de méthodes basées sur le gradient stochastique qui produisent des résultats approximatifs et d'autre part, de méthodes récursives exactes qui déterminent le modèle exact. Les premières sont simples et très rapides, tandis que les secondes sont généralement difficiles à mettre en œuvre et complexes en coût de calcul. Ces méthodes récursives permettent de prendre en compte les modifications locales de classes et dans certaines mesures, d'effectuer le suivi de classes évolutives. Elles sont très utiles dans la conception d'algorithmes de classification dynamique mais ne suffisent pas à résoudre tous les problèmes en environnement non-stationnaire. En effet, des règles supplémentaires sont nécessaires pour traiter les problèmes de fusion et de scission de classes.

3.4 Fusion et scission de classes

Dans la littérature, il existe très peu d'algorithmes de classification dotés de procédures de fusion et de scission. La plupart des algorithmes proposés conviennent particulièrement aux problèmes de classification non-supervisée et nécessitent la connaissance *a priori* de toutes les données d'apprentissage. Les premières idées de fusion et de scission de classes ont été initialement introduites dans les problèmes de traitement de signal [Green, 1995; Richardson & Green, 1997]. En particulier, Richardson & Green [1997] développent des opérations de fusion et de scission pour construire l'algorithme *Reversible Jump Markov Chain Monte Carlo* (RJMCMC). Cet algorithme est basé sur les méthodes MCMC et les règles bayésiennes mais utilise un modèle mélange gaussien unidimensionnel. L'algorithme RJMCMC ne nécessite pas la connaissance a priori du nombre de composants du modèle. Bien que les méthodes MCMC trouvent en principe l'optimum global, ils ont l'inconvénient majeur d'introduire un coût de calcul trop important. D'ailleurs, dans la plupart des problèmes de classification, la complexité des algorithmes basés sur le MCMC dépasse celle de algorithmes EM. Il existe des variantes de l'algorithme RJMCMC qui étendent son utilisation aux données multidimensionnelles [Stephens, 2000b; Nobile & Fearnside, 2005]. Mais, ces algorithmes ne peuvent fonctionner qu'en environnement fermé et ne résolvent pas les problèmes de complexité du RJMCMC.

L'algorithme Competitive-EM que nous allons détailler dans la section suivante recherche la convergence globale de l'EM à travers des mécanismes de fusion et de scission de classes gaussiennes multidimensionnelles. Il est basé sur le même principe du RJMCMC mais la quantité de calculs effectués est plus faible.

3.4.1 Competitive-EM : fusion et scission de modèles gaussiens

Nous précisions tout d'abord que le Competitive-EM n'est pas un algorithme récursif, il est présenté dans ce chapitre pour ses règles de fusion et de scission qui sont des techniques de modélisation adaptive d'une partition dynamique. Cet algorithme a été proposé par Zhang & al. [2003, 2004] afin de pallier aux problèmes de convergence locale de l'algorithme EM [Govaert, 2003]. Rappelons que les deux principaux inconvénients de l'algorithme EM sont : la convergence vers les optima locaux qui dépendent de l'initialisation de l'algorithme et la nécessité de connaître *a priori* le nombre de classes. Le Competitive-EM est construit avec les procédures basiques de l'EM (Expectation et Maximisation) et est complété avec des critères de détection de convergence locale et de procédures de fusion et de scission [Zhang & al., 2004]. Grâce à son processus d'apprentissage compétitif, cet algorithme se présente comme une version améliorée de l'EM classique. Il a deux avantages considérables :

- Le Competitive-EM est moins sensible à l'initialisation et au bruit dans les données contrairement à l'algorithme EM classique sur lequel il s'appuie.
- Lorsque l'algorithme converge vers un optimum local, les classes mal formées sont fusionnées ou scindées grâce à son processus d'apprentissage compétitif.

On considére un ensemble d'apprentissage $\{X_1,...,X_L\}$ à partir duquel l'on souhaite déterminer la partition optimale en utilisant un modèle de mélange gaussien de paramètre $\Theta=\{\Theta_1,...,\Theta_M\}$. Rappelons que chaque composante du mélange est caractérisée par les paramètres moyenne $\bar{X}_m$ et matrice de covariance Σ_m et pondérée dans le mélange par sa proportion π_m. Les paramètres du modèle et les classes d'appartenance des données sont *a priori* inconnus. Les paramètres du modèle de mélange sont tout d'abord estimés à l'aide de l'algorithme EM qui recherche la maximisation du critère de vraisemblance. L'EM converge le plus souvent vers un maximum local de ce critère. A la fin de l'estimation des paramètres du modèle par l'algorithme EM, certaines classes de partition sont surpeuplées ou adjacentes (chevauchement). L'idée du Competitive-EM consiste dans un premier temps à détecter ces classes présentant des défauts et dans un second temps à recalculer les paramètres à l'aide d'opérations de fusion et de scission de façon à atteindre l'optimum global. Cette idée fût d'abord utilisée dans l'algorithme SMEM (Split & Merge EM) [Ueda & al., 1998]. Cependant, les mécanismes de fusion et de scission proposés dans le SMEM n'ont pas de justification théorique et l'estimation des paramètres se fait de façon indépendante. Avant de présenter les mécanismes de fusion et de scission du Competitive-EM, étudions d'abord ses critères de détection.

3.4.1.1 Critères de détection de convergence locale

Ces critères de détection de convergence locale recherchent d'abord les régions de l'espace dans lesquelles subsistent des problèmes d'optima locaux, et ensuite sélectionnent le mécanisme de fusion ou de scission adapté pour traiter ces problèmes. Le critère de fusion est défini en évaluant la corrélation entre les composantes du mélange deux par deux tandis que celui de scission est établi en utilisant la divergence de Kullback-Leibler [Zhang & al. 2003].

Considérons Φ_m et $\Phi_{m'}$ les paramètres de deux composantes quelconques du mélange, on définit un coefficient de corrélation $J_{merg}(\Phi_m,\Phi_{m'})$ tel que :

$$J_{merg}(\Phi_m,\Phi_{m'})=\frac{P_m(\Phi)^T P_{m'}(\Phi)}{\|P_m(\Phi)\|\|P_{m'}(\Phi)\|} \qquad 3.45$$

où $P_m(\Phi)=[P(m,X_1;\Phi),...,P(m,X_L;\Phi)]^T$ est un vecteur constitué des probabilités a posteriori :

$$P(m|X_i,\Phi)=\pi_m\varphi(X_i,\Phi_m)\Big/\sum_{j=1}^{M}\pi_j\varphi(X_i,\Phi_j) \qquad 3.46$$

Le critère de fusion est défini à partir du coefficient de corrélation (3.45) en remarquant que si $J_{merg}(\Phi_m,\Phi_{m'})\to 1$, les composantes m et m' sont fortement corrélées et doivent donc être fusionnées; tandis que si $J_{merg}(\Phi_m,\Phi_{m'})\to 0$, elles sont orthogonales, c'est-à-dire bien séparées.

Pour la définition du critère de scission de classe, la divergence de Kullback-Leibler est définie pour une composante m de paramètre Φ_m de la manière suivante :

$$J_{split}(\Phi_m,\Phi)=\int_{\chi} g_m(X,\Phi)\log\frac{g_m(X,\Phi)}{\varphi(X,\Phi_m)}.dX \qquad 3.47$$

$g_m(X,\Phi)$ désigne la densité locale de données autour de la composante m et définit telle que :

$$g_m(X,\Phi)=\sum_{i=1}^{L}\delta(X-X_i)P(m|X_i,\Phi)\Big/\sum_{i=1}^{L}P(m|X_i,\Phi) \qquad 3.48$$

avec $\delta(X-X_i)=1$ si X_i est proche de la composante m et $\delta(X-X_i)=0$, sinon.

Un seuil choisi de façon expérimentale est fixé sur J_{split} pour la détection de scission.

3.4.1.2 Opérations de fusion et de scission

Supposons que le critère de détection de convergence locale (3.45) a détecté deux composantes m et m' candidates à la fusion. Zhang & al. [2003] déterminent les opérations de fusion en montrant qu'il existe une relation entre les paramètres de la composante m^* résultante de la fusion et les paramètres des composantes m et m' à travers les formules :

$$\pi_{m^*}=\pi_m+\pi_{m'} \qquad 3.49$$

$$\bar{X}_{m^*}=\frac{1}{\pi_{m^*}}\left(\pi_m\bar{X}_m+\pi_{m'}\bar{X}_{m'}\right) \qquad 3.50$$

$$\Sigma_{m^*}=\frac{1}{\pi_{m^*}}\left(\pi_m\Sigma_m+\pi_{m'}\Sigma_{m'}+\pi_m\bar{X}_m\bar{X}_m^T+\pi_{m'}\bar{X}_{m'}\bar{X}_{m'}^T\right)-\bar{X}_{m^*}\bar{X}_{m^*}^T \qquad 3.51$$

Comparées à celles utilisées dans l'algorithme SMEM [Ueda & al., 1998], ces formules donnent une meilleure estimation de la fusion en conservant une relation de dépendance entre les paramètres de la composante.

La scission de classe est une opération inverse de la fusion. On en déduit donc que les paramètres des composantes résultantes de la scission doivent satisfaire aux équations (3.49), (3.50) et (3.51). Mais contrairement à la fusion, la scission de classe est un problème mal-posé [Tikhonov & Arsenin, 1977], c'est-à-dire que le nombre des paramètres à estimer est supérieur au nombre d'équations. En faisant l'hypothèse que la matrice de covariance est symétrique et définie positive, Zhang & al. [2003] propose un opérateur de scission basé sur la méthode de *décomposition en valeurs singulières*. Ainsi, lorsque le critère (3.48) détecte une composante m candidate à la scission, en supposant p^* et q^* les composantes résultantes de la scission de m, l'opérateur de scission est défini à l'aide du théorème suivant :

> **Théorème 3.1** [Golub & Loan, 1996] : Si la matrice Σ est symétrique et définie positive, alors il existe une matrice $A=[a_1,...,a_D]$ de telle sorte que : $\Sigma=AA^T$, avec $a_d a_{d'}^T=\lambda_d$ si $i=j$ et $a_i a_j^T=0$ sinon.

Les valeurs propres λ_d sont ordonnées suivant un ordre décroissant : $\lambda_1\geq,...,\geq\lambda_d$. En posant $\Sigma_m=A_mA_m^T$, $\Sigma_{p^*}=A_{p^*}A_{p^*}^T$, $\Sigma_q=A_{q^*}A_{q^*}^T$, les paramètres de p^* et q^* sont estimés tels que :

$$\pi_{p^*} = \alpha\pi_m \; ; \qquad \pi_{q^*} = (1-\alpha)\pi_m \tag{3.52}$$

$$\bar{X}_{p^*} = \bar{X}_m - \sqrt{\frac{\pi_{j^*}}{\pi_{p^*}}}u.a_d^{(m)} \; ; \qquad \bar{X}_{q^*} = \bar{X}_m - \sqrt{\frac{\pi_{p^*}}{\pi_{q^*}}}u.a_d^{(m)} \tag{3.53}$$

$$a_s^{(p^*)} = \begin{cases} \sqrt{\beta(1-u^2)\frac{\pi_m}{\pi_{p^*}}}a_s^{(m)}, & s = d \\ \sqrt{\frac{\pi_{q^*}}{\pi_{p^*}}}a_s^{(m)}, & s \neq d \end{cases} \; ; \qquad a_s^{(q^*)} = \begin{cases} \sqrt{(1-\beta)(1-u^2)\frac{\pi_m}{\pi_{q^*}}}a_s^{(m)}, & s = d \\ \sqrt{\frac{\pi_{p^*}}{\pi_{q^*}}}a_s^{(m)}, & s \neq d \end{cases}$$

$d \in [1,...,D]$, avec D la dimension de l'espace et $\alpha, u, \beta \in [0,1]$ des coefficients libres.

$$\begin{aligned} \Sigma_{p^*} &= \frac{\pi_{q^*}}{\pi_{p^*}}\Sigma_m + \left(\beta u^2 - \beta - u^2\right)\frac{\pi_m}{\pi_{p^*}}a_d^{(m)}\left(a_d^{(m)}\right)^T + a_d^{(m)}\left(a_d^{(m)}\right)^T \\ \Sigma_{q^*} &= \frac{\pi_{p^*}}{\pi_{q^*}}\Sigma_m + \left(\beta u^2 - \beta - u^2\right)\frac{\pi_m}{\pi_{q^*}}a_d^{(m)}\left(a_d^{(m)}\right)^T + a_d^{(m)}\left(a_d^{(m)}\right)^T \end{aligned} \tag{3.54}$$

L'estimation des paramètres de composantes à travers cette opération de scission, nécessite le réglage de trois coefficients (free parameters). Dans les expériences, Zhang & al. [2003] propose de fixer ces coefficients comme suit : $\alpha = u = \beta = 0,5$.

3.4.2 Analyse du Competitive-EM

Le Competitive-EM est une amélioration de l'algorithme EM dont la qualité est affectée par les problèmes d'initialisation et de convergence locale. Dans le processus d'apprentissage du Competitive-EM, les critères de détection de convergence locale aident à trouver les régions d'optima locaux en utilisant une mesure de corrélation entre les classes (3.45) et la divergence de Kullback-Leibler (3.47). Les seuils de détection sont choisis de façon expérimentale. Un bon choix de ces seuils est une condition nécessaire pour la convergence vers l'optimum global. Dans le cas du SMEM, les paramètres des composantes concernées par la fusion ou la scission sont plutôt réinitialisés de façon approximative et identique. Concernant le Competitive-EM, les formules (3.49) à (3.51) de la fusion calculent le modèle exact tandis que les formules (3.52) à (3.54) de la scission ne peuvent donner que des modèles approchés. Ceci est dû au fait que la scission de classe fait partie des problèmes dits mal-posés (*ill-posed problems*) dont la résolution nécessite l'introduction de paramètres libres. En terme de complexité, il est clair que l'algorithme Competitive-EM effectue plus de calculs que l'EM classique, mais il a un coût de calcul moins important que le RJMCMC. L'utilisation du Competitive-EM n'est pas envisageable dans les problèmes de classification dynamique à cause d'une part de sa grande complexité algorithmique et d'autre part, de l'utilisation des probabilités (univers clos). Néanmoins, cet algorithme a des bonnes propriétés de convergence dans un contexte de classification non-supervisé et hors-ligne.

3.5 Conclusion

Dans ce chapitre, les techniques de modélisation adaptative ont été abordées dans l'objectif d'apporter quelques outils servant à la conception d'algorithmes de classification dynamique. Nous avons exposé deux catégories de méthodes récursives proposées pour l'apprentissage incrémental et quelques mécanismes de fusion et de scission introduits pour traiter les problèmes locaux. Les aptitudes de ces méthodes pour la modélisation adaptative de modèles de mélange ou des fonctions d'apprentissage à noyau ont été montrées.

Parmi les méthodes récursives étudiées, la première catégorie est celle qui regroupe les méthodes basées sur le gradient stochastique. L'algorithme du gradient stochastique s'applique aux problèmes d'apprentissage séquentiel dans lequel l'on souhaite estimer des modèles successifs en minimisant ou en maximisant un critère global. La minimisation d'un tel critère est effectuée en appliquant la technique de la descente du gradient tandis que sa maximisation est obtenue en utilisant la technique de la montée du gradient. Le gradient stochastique est une méthode simple et rapide, très utilisée dans les applications en ligne. Ses capacités de mise à jour itérative de modèles de classification ont été montrées à travers les algorithmes incrémentaux de *k-means*, CEM et le NORMA. Le gradient stochastique converge mais la convergence vers l'optimum global n'est pas garantie. Cette méthode produit des modèles approximatifs qui restent tout de même satisfaisants pour la plupart des problèmes. Les méthodes de mise à jour exactes constituent la seconde catégorie. Ce sont des méthodes convergentes. Leur mise en œuvre est une épreuve quelquefois difficile. Dans ce chapitre, quelques méthodes récursives exactes ont été exposées pour la mise à jour du modèle gaussien et l'estimation itérative de densité à l'aide de méthodes à noyau. Les méthodes récursives exactes sont quelquefois limitées dans les problèmes temps-réel à cause de leur complexité algorithmique trop élevée.

Par ailleurs, dans la dernière partie de ce chapitre, nous avons présenté un mécanisme de fusion et de scission de classes à travers l'étude du Competitive-EM. Développé pour améliorer la convergence de l'algorithme EM, cet algorithme dispose de critères de détection de convergence locale et des opérations de scission et de scission de classes. Les critères de détection sont basés sur une mesure de corrélation et la divergence de Kullback-Leibler. Ces critères sélectionnent l'opération à effectuer pour résoudre certains problèmes d'optima locaux. Les opérations de fusion et de scission du Competitive-EM estiment les modèles optimaux à partir des classes comportant de défauts dans la modélisation avec le mélange gaussien. Cependant, l'utilisation de cet algorithme est réservée à la classification supervisée avec la connaissance *a priori* de toutes les données. En effet, il se base sur une modélisation probabiliste dans un univers fermé, il ne convient donc pas à l'apprentissage en ligne où la base de données est incomplète. Par ailleurs, les mécanismes de fusion et de scission introduisent un coût de calcul trop important à cause de l'analyse effectuée à chaque itération sur toutes les classes pour la détection de défauts de modélisation.

Après l'étude menée dans le chapitre 2 qui a montré les performances des modèles de mélange ou des fonctions d'apprentissage à noyau, nous venons de présenter dans le chapitre 3 quelques techniques de modélisation adaptative pour la classification de données spatio-temporelles (ou non-stationnaires). Ces deux chapitres, largement consacrés à l'étude bibliographique, ont ainsi apporté des outils de modélisation et des techniques de mise à jour récursive intéressants pour la conception d'algorithmes de classification dynamique. En se basant sur la méthode générique décrite dans la partie 1.4 du chapitre 1, nous proposons dans le chapitre suivant deux algorithmes de classification dynamique.

Chap. 4 : Algorithmes de classification dynamique

4.1 Introduction

Le travail mené jusqu'à présent dans ce mémoire, a montré les difficultés liées à la classification dynamique. L'étude bibliographique a permis de présenter quelques méthodes possédant des propriétés intéressantes pour la classification dynamique. Le présent chapitre a pour but d'étudier deux algorithmes que nous avons développés suivant la description générique de classifieurs dynamiques proposée au chapitre 1 (partie 4.4).

Rappelons tout d'abord les principales difficultés posées par la problématique de la classification de données non-stationnaires (Définition 1.1). L'apprentissage se fait de façon séquentielle. Il s'agit de problèmes de classification en ligne qui nécessitent la mise en œuvre de règles d'apprentissage incrémental. Par ailleurs, on ne dispose d'aucune connaissance *a priori* sur les étiquettes des données, ni sur le nombre de classes. La classification se fait en mode non-supervisé. A ces difficultés s'ajoute l'évolution temporelle de classes due à la non-stationnarité des données. Une modélisation continue et adaptative de classes est donc nécessaire. Cette modélisation doit permettre le suivi de classes évolutives. De même, la non-stationnarité des données et l'évolution des classes se caractérisent par divers phénomènes dynamiques comme l'apparition des classes, l'évolution de classes, etc. Afin de prendre en compte ces phénomènes, le classifieur dynamique doit être doté de règles d'auto-adaptation.

Les réseaux de neurones, plus particulièrement les réseaux à architectures évolutives ont de grandes capacités de classification en environnement non-stationnaire. Ces capacités sont prouvées à travers l'étude des algorithmes CDL, FMM et ESOM (chapitre 1, partie 1.3). Grâce à leurs principes de création et d'adaptation de neurones, ces algorithmes disposent de capacités d'apprentissage et d'adaptation. Malheureusement, ils sont victimes de problèmes d'optima locaux à cause de leurs modèles de classification. Ces modèles (hyper-cubiques, hyper-sphériques, ...) sont généralement des approximations grossières.

L'étude menée dans le chapitre 2 a montré que les modèles de mélange et les fonctions d'apprentissage à noyau sont des modèles statistiques ayant de bonnes performances en

classification. Le modèle de mélange gaussien est un modèle paramétrique donnant des résultats satisfaisants dans la plupart des applications. Quant aux fonctions d'apprentissage à noyau, ce sont des modèles non-paramétriques capables de modéliser des formes complexes. En revanche, les méthodes de classification utilisant ces modèles ne sont pas dotées de règles d'auto-adaptation nécessaires en environnement dynamique. Cette lacune nous a conduit à étudier quelques méthodes récursives de modélisation continue et adaptative (chapitre 3).

Dans le présent chapitre, nous présentons deux algorithmes de classification dynamique. Le processus d'apprentissage de ces algorithmes est illustré sur la figure 4.1.

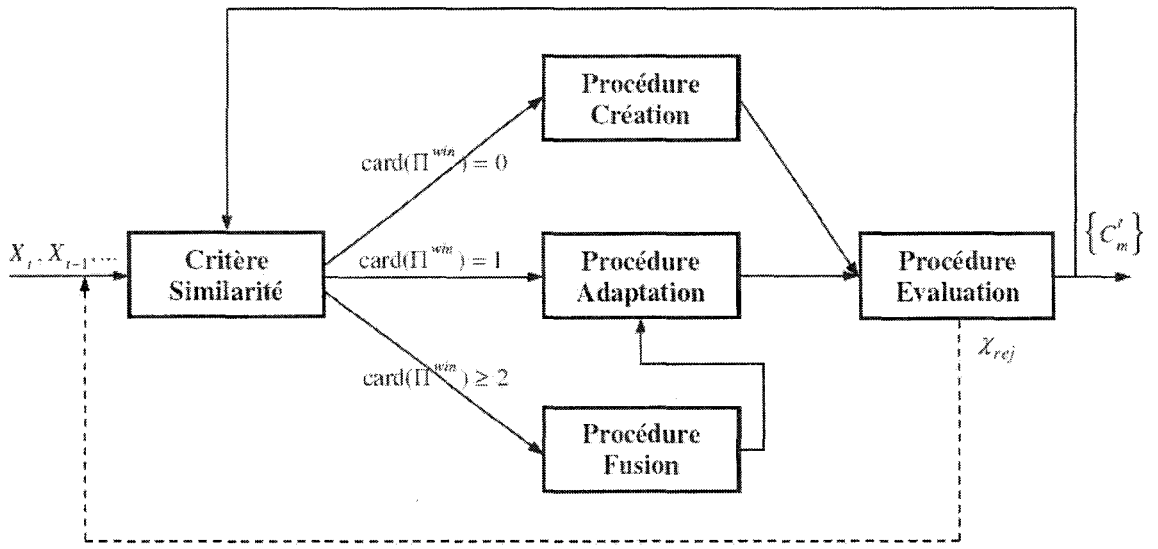

Figure 4.1: Processus d'apprentissage d'algorithmes de classification dynamique

Ce processus conçu pour un fonctionnement en ligne, est élaboré avec un critère de similarité et quatre procédures d'apprentissage : **procédure de création, procédure d'adaptation, procédure de fusion et procédure d'évaluation**. A chaque fois qu'une nouvelle donnée se présente, le critère de similarité permet de choisir la procédure adéquate pour la mise à jour du modèle de classification. Ensuite, la procédure d'évaluation analyse la représentation des classes créées. Cette procédure est constituée d'un mécanisme de scission de classes et d'un mécanisme d'élimination de classes parasites. Après l'élimination d'une classe, les données de cette classe sont stockées dans un ensemble de rejet X_{rej}. Ces données sont à nouveau présentées au processus d'apprentissage sauf les plus vieilles qui sont éliminées définitivement. La difficulté majeure de la conception des algorithmes réside dans la définition d'un critère de similarité adéquat et la mise en œuvre des procédures d'apprentissage à partir du modèle choisi. Ces procédures doivent être élaborées dans l'objectif de conférer à l'algorithme des capacités d'auto-adaptation.

Le présent chapitre est organisé en deux parties principales :

- La première partie présente l'**AUDyC** (*Auto-Adaptive & Dynamical Clustering*). Cet algorithme est basé sur une technique de modélisation inspirée du modèle de mélange gaussien. Les classes y sont décrites suivant une approche multimodale.
- La seconde partie est consacrée à l'étude de l'algorithme **SAKM** (*Self-Adaptive Kernel Machine*). Cet algorithme modélise les classes de la partition dynamique en s'appuyant sur les fonctions d'apprentissage à noyau.

Pour chacun de ces algorithmes, on présente d'abord le modèle de classification. Ensuite, un critère de similarité est défini spécifiquement à chaque modèle de classification. Les différentes procédures d'apprentissage sont ensuite détaillées. Nous terminons par l'étude de la convergence et la complexité de chacun des deux algorithmes.

4.2 AUto-adaptive & Dynamical Clustering : nouvelle version

L'algorithme AUDyC est introduit sous sa première version dans les travaux de thèse de Lurette [2003]. Dans la présente partie, nous proposons une nouvelle version de cet algorithme avec un formalisme plus théorique et des règles d'auto-adaptation plus élaborées. Développé à partir de la description générique que nous avons proposée, cet algorithme s'inspire du modèle de mélange gaussien décrit suivant l'approche multimodale. Un critère de vraisemblance est introduit pour évaluer la qualité de classification de l'AUDyC. Par ailleurs, nous apportons à l'algorithme des améliorations importantes qui visent à pallier les limitations constatées dans la première version. Ces améliorations concernent principalement le processus d'apprentissage. Il s'agit plus précisément de l'élaboration des nouvelles procédures de fusion et de scission de classes.

Dans la section suivante, nous décrivons la technique de modélisation adoptée pour l'algorithme AUDyC. Ensuite dans la section 4.2.2, le critère de similarité de l'algorithme est défini à partir d'une fonction d'appartenance. Les procédures d'apprentissage sont exposées dans la section 4.2.3. Enfin, la section 4.2.4 étudie la convergence et la complexité de l'algorithme avant de terminer cette partie avec une discussion.

4.2.1 Modèle de classification de l'AUDyC

Le modèle de classification de l'AUDyC est un modèle de mélange fortement inspiré du **mélange gaussien**. Il est décrit en utilisant une technique de **modélisation possibiliste** qui convient particulièrement aux problèmes d'apprentissage en ligne. Très utilisé en Reconnaissance de Formes (RdF), le modèle gaussien convient à l'approximation de la plupart des distributions de données. Par ailleurs, c'est un outil statistique dont les éléments mathématiques sont relativement bien maîtrisés. Afin de lui conférer des capacités de modélisation élevées en environnement de données non-stationnaires, le modèle de classification de l'AUDyC est construit suivant l'**approche multimodale**. A travers cette

approche, une classe complexe est modélisée par la combinaison de plusieurs prototypes hyper-elliptiques dont chacun s'adapte localement à la distribution de données : on parle également d'approche multi-prototype. Suivant cette description, chaque classe C_m de la partition dynamique est représentée par une fonction ψ_m de paramètre Θ_m^t. Cette fonction caractérise le mélange de densités au sein de la classe C_m et est définie :

$$\psi_m(X) = \sum_{j=1}^{J_m} \varphi\left(X, \theta_{j,m}^t\right) \qquad 4.1$$

où J_m est le nombre de prototypes de la classe C_m. $\varphi\left(\bullet, \theta_{j,m}^t\right)$ représente la fonction de densité locale du prototype $P_{j,m} \subseteq C_m$. Elle est définie à chaque instant t par l'expression suivante :

$$\varphi\left(X, \theta_{j,m}^t\right) = p_{j,m} \times \exp\left(-\frac{1}{2}\left(X - \bar{X}_{j,m}^t\right)^T \left(\Sigma_{j,m}^t\right)^{-1} \left(X - \bar{X}_{j,m}^t\right)\right) \qquad 4.2$$

Les $p_{j,m}$ sont les poids (ou possibilités) respectifs des prototypes $P_{j,m}$ dans la classe C_m. $\bar{X}_{j,m}^t$ et $\Sigma_{j,m}^t$ sont respectivement la moyenne et la matrice de covariance du prototype $P_{j,m}$. Elles constituent le paramètre $\theta_{j,m}^t$ de ce prototype. De cette façon, la fonction ψ_m modélisant la classe C_m a pour paramètre le vecteur $\Theta_m^t = \left[\theta_{1,m}^t, \ldots, \theta_{j,m}^t, \ldots, \theta_{J_m,m}^t\right]$.

Suivant la description du modèle de classification de l'AUDyC, chaque classe C_m est définie par l'ensemble des données :

$$C_m = \left\{ X \in \chi \Big/ \sum_{j=1}^{J_m} \varphi\left(X, \theta_{j,m}^t\right) - \mu_1 \geq 0 \right\} \qquad 4.3$$

Et, un prototype $P_{j,m} \subseteq C_m$ correspond à l'ensemble des données :

$$P_{j,m} = \left\{ X \in \chi \Big/ \varphi\left(X, \theta_{j,m}^t\right) - \mu_2 \geq 0 \right\} \qquad 4.4$$

La condition $\mu_1 \leq \mu_2$ est imposée pour satisfaire la relation $P_j \subseteq C_m$. Si $\mu_1 = \mu_2$, la classe ne peut contenir que l'ensemble des données présentes dans ses prototypes. Si $\mu_1 < \mu_2$, la classe peut contenir d'autres données que celles présentes de ses prototypes. Les paramètres μ_1 et μ_2 déterminent l'hyper-volume des classes et des prototypes. Nous reviendrons sur les paramètres μ_1 et μ_2 dans la section 4.2.2.

Lien avec le modèle de mélange gaussien : transformation possibilité-probabilité

En introduisant une transformation possibilité-probabilité, on peut facilement retrouver la description du modèle de mélange gaussien classique. Suivant la modélisation probabiliste et l'approche multimodale, le mélange gaussien se définit à partir de l'équation (4.2), en utilisant l'expression de la fonction de densité gaussienne. Celle-ci s'écrit :

$$\varphi\left(X,\theta_{j,m}^{t}\right)=\frac{\pi_{j,m}}{(2\pi)^{1/2}\left(\det\left(\Sigma_{j,m}^{t}\right)\right)^{1/2}}\times\exp\left(-\frac{1}{2}\left(X-\bar{X}_{j,m}^{t}\right)^{T}\left(\Sigma_{j,m}^{t}\right)^{-1}\left(X-\bar{X}_{j,m}^{t}\right)\right) \qquad 4.5$$

$\pi_{j,m}$ est la proportion (ou probabilité *a priori*) du $P_{j,m}$ dans la classe C_m.

De (4.2) et (4.5), on établit la relation entre les poids $p_{j,m}$ et les proportions $\pi_{j,m}$ telle que :

$$p_{j,m}=\frac{\pi_{j,m}}{(2\pi)^{1/2}\left(\det\left(\Sigma_{j,m}^{t}\right)\right)^{1/2}} \qquad 4.6$$

Et à partir de la relation (4.6), on peut revenir à la modélisation probabiliste du mélange gaussien. Cependant, à cause de la contrainte de normalisation sur des probabilités *a priori*, nous préférons la modélisation possibiliste qui est mieux adaptée aux problèmes d'apprentissage dans un univers de discours ouvert. Par ailleurs, l'utilisation de la relation (4.6) n'est pas sans condition. En effet, le passage des probabilités aux possibilités nécessite la satisfaction de la contrainte sur les possibilités : $0\le p_{j,m}\le 1$ [Mouchaweh, 2002]. Et inversement, le passage des possibilités en probabilités n'est possible que sous les conditions : $0\le\pi_{j,m}\le 1$ et $\sum_{j,m}\pi_{j,m}=1$. En réalité, dans les méthodes probabilistes, les probabilités $\pi_{j,m}$ sont directement estimées avec les paramètres du modèle au cours de la classification des données. Tandis que dans la modélisation possibiliste, les possibilités $p_{j,m}$ sont calculées à partir de transformation probabilité-possibilité ou sont choisies selon certaines considérations sur le problème [Mouchaweh, 2002]. Par exemple, dans le cadre de la supervision des processus industriels, on peut choisir d'attribuer des possibilités plus ou moins fortes aux modes (classes) de défaillances en fonction de leurs criticités. Dans le cadre de notre étude sur la classification dynamique, nous adaptons l'hypothèse simplificatrice qui consiste à utiliser des possibilités calculées pour les prototypes d'une même classe de la manière suivante :

$$p_{j,m}=\frac{\mathrm{card}(P_{j,m})}{\mathrm{card}(C_m)},\ P_{j,m}\subseteq C_m \quad\text{et}\quad p_{j,m}=0,\ \text{sinon;}\quad \mathrm{card}(\bullet)\text{: cardinalité} \qquad 4.7$$

La possibilité $p_{j,m}$ correspond ainsi au poids en terme de population du prototype $P_{j,m}$ dans la classe C_m. On en déduit de l'expression (4.7) que la somme des possibilités des prototypes d'une même classe est inférieure ou égale à 1. Par contre, la somme de toutes les possibilités n'est pas bornée (contrairement aux probabilités *a priori*). Cette propriété est intéressante car elle traduit le fait que l'espace de classification est ouvert à la création de nouveaux prototypes et classes.

Le modèle de classification de l'AUDyC correspond à un mélange gaussien parcimonieux [Govaert, 2003]. La figure 4.2 illustre la modélisation des classes de l'AUDyC avec des possibilités unitaires. Rappelons que les classes sont susceptibles d'évoluer dans le cas de données non-stationnaires. La modélisation adaptative et le suivi de classes évolutives sont rendus possible grâce aux procédures d'apprentissage (section 4.2.3).

Vraisemblance possibiliste

Afin de quantifier en ligne la qualité de la classification de l'AUDyC, une fonction objectif est définie. Rappelons que la plupart des méthodes de classification basées sur des modèles de mélanges recherche l'optimisation d'un critère de vraisemblance (Chapitre 2, Section 2.2.3.1). C'est le cas notamment pour l'algorithme très connu EM (Expectation-Maximisation). Cependant, l'utilisation d'un tel critère n'est possible que dans un univers clos où toutes les hypothèses sont connues *a priori*. En adoptant la modélisation possibiliste, nous proposons d'introduire un critère de vraisemblance possibiliste L_{poss} qui mesure la qualité du modèle de classification de l'AUDyC. L'objectif consiste donc à maximiser ce critère exprimé à chaque instant t par :

$$L_{poss}\left[\Im^{(t)}\right]=\frac{1}{J}\sum_{m=1}^{M}\sum_{j=1}^{J_m}\sum_{X_t\in P_{j,m}}\log\left(p_{j,m}\times\exp\left(-\frac{1}{2}\left(X-\bar{X}_{j,m}^{t}\right)^{T}\left(\Sigma_{j,m}^{t}\right)^{-1}\left(X-\bar{X}_{j,m}^{t}\right)\right)\right) \qquad 4.8$$

où J est le nombre de prototypes constituant la partition dynamique.

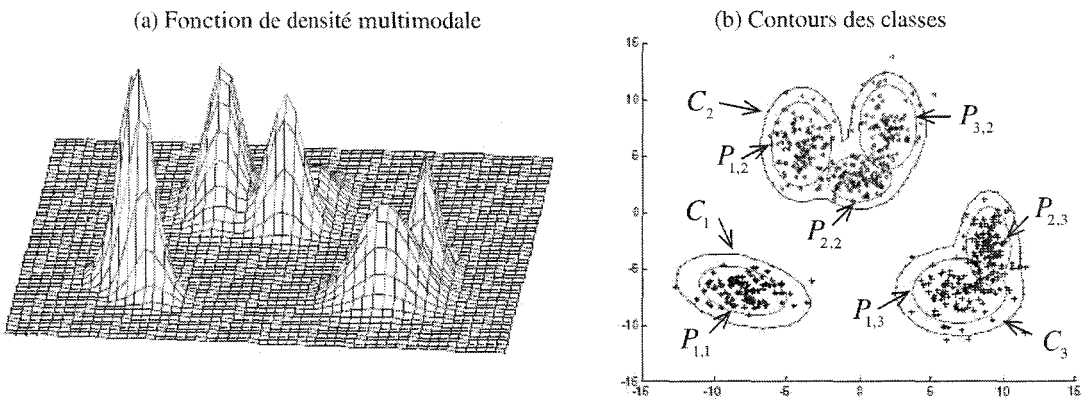

Figure 4.2 : Mélange gaussien, approche multimodale. (a) Représentation 3D de la fonction de densité des données. (b) La partition est décrite par : $C_1=\{P_{1,1}\}, C_2=\{P_{1,2},P_{2,2},P_{3,2}\}$, $C_3=\{P_{1,3},P_{2,3}\}$.

Après la définition du modèle de classification de l'AUDyC et l'introduction d'une fonction objectif, nous allons proposer un critère de similarité avant de détailler les procédures d'apprentissage de l'algorithme.

4.2.2 Critère de similarité : Fonction d'appartenance

La mesure de similarité entre chaque nouvelle donnée X_t et les classes C_m déjà créées est définie en utilisant une fonction d'appartenance. Cette fonction notée $\mu_{j,m}$, mesure le degré de d'appartenance de la donnée X_t avec chaque prototype gaussien $P_{j,m}\subseteq C_m$. Elle est définie de la manière suivante :

$$\mu_{j,m}(X_t) = \exp\left(-\frac{1}{2}\left(d\left(X, P_{j,m}\right)\right)\right) = \exp\left(-\frac{1}{2}\left(X_t - \bar{X}^t_{j,m}\right)^T \left(\Sigma^t_{j,m}\right)^{-1}\left(X_t - \bar{X}^t_{j,m}\right)\right) \qquad 4.9$$

où $d\left(X, P_{j,m}\right)$ est la distance de Mahanalobis. Cette distance tient compte de la dispersion des données dans toutes les directions de la zone d'influence du prototype.

La fonction d'appartenance ainsi définie possède les propriétés suivantes :

1. $\forall t, \forall j, \forall m, \quad 0 < \mu_{j,m}(X_t) \leq 1$

2. $\forall t, \forall j, \forall m, \quad 0 < \sum_{k=1}^{t} \mu_{j,m}(X_k) < t$

Suivant la modélisation multimodale de l'AUDyC, la mesure de similarité entre une donnée X_t et la classe est définie à chaque instant t telle que :

$$S(X_t, C_m) = \max_j\left(S(X_t, P_{j,m})\right) = \max_j\left(\mu_{j,m}(X_t)\right), \qquad j \in [1, \ldots, J_m] \qquad 4.10$$

Cette mesure est décroissante sur $\Re$ lorsque X_t s'éloigne de $P_{j,m} \subseteq C_m$ quelque soit la direction dans l'espace χ. De même, elle est bornée entre [0,1] et prend la valeur 1 si $X_t = \bar{X}^t_{j,m}$ et 0 si X_t est infiniment éloignée de la classe C_m. Par convention, nous posons : $S(X, \varnothing) = 0$ pour gérer les conditions initiales dans un espace de classification « vide ».

A partir des équations (4.9) et (4.10), on peut maintenant établir les critères de similarité donnant les ensembles Π_C^{win} et Π_P^{win} respectivement de classes et de prototypes gagnants :

$$\Pi_C^{win} = \left\{C_m / S(X, C_m) \geq \mu_{\min}\right\} \qquad 4.11$$

$$\Pi_P^{win} = \left\{P_{j,m} / \mu_{j,m}(X_t) \geq \mu_{\max}\right\} \qquad 4.12$$

$\mu_{\min}$ et $\mu_{\max}$ sont les seuils d'appartenance avec la condition $\mu_{\min} \leq \mu_{\max}$.

Remarque : En utilisant les formules (4.3) et (4.11), (4.4) et (4.12), on peut établir les relations entre les valeurs de μ_1 et μ_2 fixant les bornes limites des classes et des prototypes et les seuils d'apprentissage $\mu_{\min}$ et $\mu_{\max}$ de manière à avoir : $\mu_1 \leq p_{j,m} \cdot \mu_{\min}$ et $\mu_2 = p_{j,m} \cdot \mu_{\max}$.

4.2.3 Processus d'apprentissage de l'AUDyC

Le processus d'apprentissage de l'AUDyC est constitué des quatre procédures proposées dans la description générique, à savoir : **création, adaptation, fusion** et **évaluation**. Chacune de ces procédures d'apprentissage s'applique aux prototypes et aux classes. Le tableau 4.1 présente la règle de décision de l'AUDyC établie à partir des critères (4.11) et (4.12).

Cas 1	(a) : $\text{card}\left(\Pi_C^{win}\right)=0$ (b) : $\text{card}\left(\Pi_C^{win}\right)=1$ et $\text{card}\left(\Pi_P^{win}\right)=0$	Procédure de Création
Cas 2	(a) : $\text{card}\left(\Pi_C^{win}\right)=1$ et $\text{card}\left(\Pi_P^{win}\right)=1$	Procédure d'Adaptation
Cas 3	(a) : $\text{card}\left(\Pi_C^{win}\right)=1$ et $\text{card}\left(\Pi_P^{win}\right)\geq 2$ (b) : $\text{card}\left(\Pi_C^{win}\right)\geq 2$	Procédure de Fusion
Cas 4	Toutes les T données	Procédure d'Évaluation

Tableau 4.1 : Règles de décision et procédures d'apprentissage de l'algorithme AUDyC.

4.2.3.1 Procédure de Création : Initialisation de prototypes, Création de classes

Le cas 1 (a) du tableau 4.1 présente une situation où aucun prototype n'est suffisamment proche de la nouvelle donnée et aucune classe n'est gagnante. Dans cette situation, le classifieur dynamique crée une nouvelle classe C_{new} à partir de la donnée X_t. C_{new} représente la première classe ($new=1$) si la partition $\Pi^{(t-1)}=\varnothing$. La création de C_{new} se fait à partir de l'initialisation de son premier prototype $P_{1,new} \subseteq C_{new}$ (figure 4.2.a). Le paramètre $\theta_{1,new}^t$ du modèle $\psi_{new}(\bullet)=\varphi(\bullet,\theta_{1,new}^t)$ est initialisé à partir de la seule donnée X_t tel que:

$$\theta_{1,new}^t=\left(\bar{X}_{1,new}^t=X_t,\Sigma_{1,new}^t=\Sigma_{ini}\right) \qquad 4.13$$

Il convient de définir les coordonnées du centre du prototype au point X_t, mais il est impossible d'estimer la matrice de covariance initiale Σ_{ini} à partir de la seule donnée X_t.

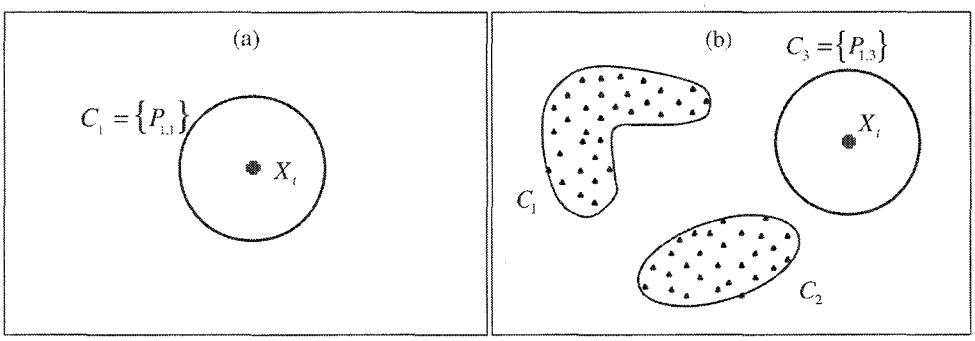

Figure 4.2 : Création de classe à la suite de l'insertion d'un prototype initialisé avec la donnée X_t comme moyenne et Σ_{ini} comme matrice de covariance initiale. (a) Création de la première classe, (b) Création d'une nouvelle classe dans une partition contenant déjà d'autres classes.

Le cas 1 (b) du tableau 4.1 correspond à une situation où la classe gagnante C_{win} contient déjà un ou plusieurs prototypes $P_{j,win}$, mais la donnée X_t n'est assez proche d'aucun de ces prototypes pour participer à sa définition (figure 4.2.b). Dans ce cas, un nouveau

prototype $P_{new,win} \subset C_{win}$ est créé avec $\varphi(\bullet, \theta^t_{new,win})$ pour fonction de densité. Le paramètre $\theta^t_{new,win}$ est initialisé de la même manière décrite par l'équation (4.13) :

$$\theta^t_{new,win} = \left(\bar{X}^t_{new,win} = X_t, \Sigma^t_{new,win} = \Sigma_{ini} \right) \qquad 4.14$$

Dans les deux situations précédemment évoquées (cas 1, tableau 4.1), nous adoptons une hypothèse simplificatrice pour l'initialisation de la matrice de covariance Σ_{ini}. Elle consiste à considérer que la zone d'influence du nouveau prototype est hyper-sphérique. Ce qui se traduit par : $\Sigma_{ini} = \sigma^2_{ini} . I_D$, où σ_{ini} est l'écart-type identique dans toutes les directions de χ et I_D est la matrice identité en D dimension. Nous proposons une méthode d'initialisation de Σ_{ini} dans l'Annexe A.4.

4.2.3.2 Procédure d'Adaptation : Mise à jour du prototype gaussien

Dans le cas 2 du tableau 4.1, une seule classe C_{win} est gagnante. La nouvelle donnée X_t est suffisamment proche d'un et un seul prototype $P_{win,win} \subseteq C_{win}$. La mise à jour de la classe C_{win} se fait alors à travers l'adaptation du prototype $P_{win,win}$ (figure 4.3). Ceci revient à adapter récursivement le paramètre $\theta^t_{win,win}$ constitué de la moyenne et de la matrice de covariance de ce prototype. La procédure d'adaptation de l'AUDyC est établie en utilisant la règle de mise à jour exacte du modèle gaussien exposée au chapitre 3, section 3.3.2.1. En supposant N_p la taille de la fenêtre de définition des prototypes et $n_{win,win}$ la cardinalité du prototype $P_{win,win}$ à l'instant $t-1$, les formules récursives sont établies comme suit :

- Si $n_{win,win} < N_p$: ajout d'information

$$\bar{X}^t_{win,win} = \bar{X}^{t-1}_{win,win} + \frac{1}{n_{win,win}+1}\left(X_t - \bar{X}^{t-1}_{win,win} \right) \qquad 4.15$$

$$\Sigma^t_{win,win} = \frac{n_{win,win}-1}{n_{win,win}} \Sigma^{t-1}_{win,win} + \frac{1}{n_{win,win}+1}\left(X_t - \bar{X}^{t-1}_{win,win} \right) \bullet \left(X_t - \bar{X}^{t-1}_{win,win} \right)^T \qquad 4.16$$

- Sinon ($n_{win,win} \geq N_p$) : ajout et retrait d'information

$$\bar{X}^t_{win,win} = \bar{X}^{t-1}_{win,win} + \frac{1}{N_p}\left(X_t - X_{t-N_p+1} \right) \qquad 4.17$$

$$\Sigma^t_{win,win} = \Sigma^{t-1}_{win,win} + \Delta X \begin{bmatrix} \frac{1}{N_p} & \frac{1}{N_p(N_p-1)} \\ \frac{1}{N_p(N_p-1)} & \frac{-(N_p+1)}{N_p(N_p-1)} \end{bmatrix} \Delta X^T \qquad 4.18$$

avec $\Delta X = \left[X_t - \bar{X}^{t-1}_{win,win},\ X_{t-N_p+1} - \bar{X}^{t-1}_{win,win} \right]$

Rappelons que ces formules récursives proviennent de la détermination de la récurrence sur le modèle gaussien (Annexe A.3).

Remarque : Dans les situations d'ajout d'information, la mise à jour de la proportion $p^t_{win,win}$ du prototype gagnant est fait simplement avec la formule (4.7) en incrémentant de 1 la cardinalité de $P_{win,win}$ et de C_{win}. En revanche, on trouve $p^t_{win,win} = p^{t-1}_{win,win}$ dans les situations d'ajout et de retrait.

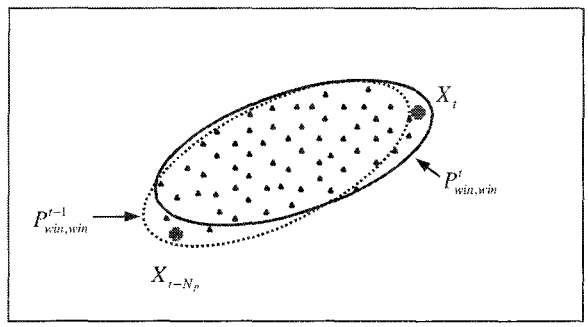

Figure 4.3 : Adaptation récursive du prototype gaussien - ajout X_t et retrait de X_{t-N_p}.

Dans la section 4.2.5.1, nous montrons que la procédure d'adaptation de l'AUDyC converge. Quelques résultats de simulation y sont donnés pour illustrer cette convergence.

4.2.3.3 Procédure de Fusion : Fusion de prototypes et Fusion de classes

La procédure de fusion proposée dans cette section est une amélioration importante apportée à l'AUDyC [Amadou & al., 2005b]. L'avantage de cette nouvelle procédure de fusion est de donner une meilleure modélisation des classes fusionnées en évitant les problèmes d'optima locaux constatés dans la version précédente de l'AUDyC.

Dans le cas 3 (a) du tableau 4.1, lorsque des données ambiguës sont partagées par deux ou plusieurs prototypes, Lecoeuche & Lurette [2003] proposent d'utiliser soit une solution d'adaptation, soit une solution de fusion. Or, chacune de ces deux solutions présentent des inconvénients. La solution d'adapter les prototypes partageant des données ambiguës entraîne un risque de chevauchement (figure 4.4.a). En revanche, la solution qui consiste à fusionner systématiquement ces prototypes occasionne des erreurs de modélisation dans certaines situations (figure 4.4.b et 4.4.c).

L'idée que nous proposons consiste à effectuer au préalable une étude des modèles des **prototypes candidats** (i.e. partageant des données ambiguës) avant de les fusionner. Cette analyse est effectuée à travers l'étude des formes et des orientations relatives des prototypes. C'est ainsi que nous proposons l'évaluation préalable d'un **critère d'acceptation** qui autorise la fusion de prototypes uniquement si celle-ci cause un défaut de modélisation. Ce critère est

basé sur l'évaluation d'une mesure de similarité A_c définie entre deux prototypes candidats $P_{win1,win}$ et $P_{win2,win}$ de la manière suivante :

$$A_c\left(\theta^t_{win1,win},\theta^t_{win2,win}\right)=tr\left(\Sigma^t_{win1,win}\bullet\Sigma^{t\ -1}_{win2,win}+\Sigma^t_{win2,win}\bullet\Sigma^{t\ -1}_{win1,win}\right)-2D \qquad 4.19$$

où $\theta^t_{win1,win}$, $\theta^t_{win2,win}$ sont les paramètres respectifs des prototypes $P_{win1,win}$ et $P_{win2,win}$.

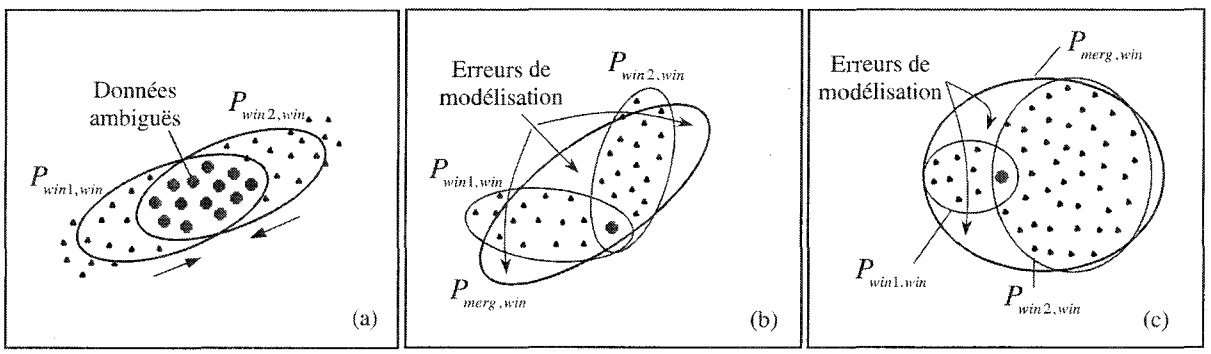

Figure 4.4 : Problématique des données ambiguës. (a) Adaptation des prototypes gagnants : chevauchement. (b) Fusion de prototypes non-alignés : erreurs de modélisation. (c) Fusion d'un gros prototype et d'un petit prototype : erreurs de modélisation.

La mesure A_c correspond en réalité au second terme de la distance de Kullback-Leibler [Shaohua & al., 2004]. Le terme A_c est sensible à la discordance locale des formes et orientations relatives des prototypes. Le critère d'acceptation de fusion est défini comme suit :

- Si $A_c \geq \varepsilon_{kl}$ (où ε_{kl} est le seuil de discordance) pour deux prototypes alors leur fusion occasionnera un problème de sous-apprentissage (figures 4.4.b et 4.4.c). Ils ne sont donc pas fusionnés, seul le prototype ayant le degré d'appartenance le plus élevé est adapté en utilisant la règle de mise à jour récursive établie dans la section 4.2.3.2.
- Sinon ($A_c < \varepsilon_{kl}$), les prototypes candidats sont fusionnés en un seul prototype $P_{merg,win}$. Le paramètre $\theta^t_{merg,win}$ du prototype résultant est calculé en utilisant la règle de fusion de prototypes gaussiens établie de la manière suivante :

$$\bar{X}^t_{merg,win}=\frac{1}{n_{merg,win}}\left(n_{win1,win}\bar{X}^t_{win1,win}+n_{win2,win}\bullet\bar{X}^t_{win2,win}\right) \qquad 4.20$$

$$\Sigma^t_{merg,win}=\frac{1}{n_{merg,win}-1}\left(\begin{array}{l}\left(n_{win1,win}-1\right)\Sigma^t_{win1,win}+\left(n_{win2,win}-1\right)\Sigma^t_{win2,win}+\\ \frac{n_{win1,win}\bullet n_{win2,win}}{n_{win1,win}+n_{win2,win}}\left(\left(\bar{X}^t_{win1,win}-\bar{X}^t_{win2,win}\right)\bullet\left(\bar{X}^t_{win1,win}-\bar{X}^t_{win2,win}\right)^T\right)\end{array}\right) \qquad 4.21$$

avec $n_{merg,win}=n_{win1,win}+n_{win2,win}$ où $n_{win1,win}=\mathrm{card}\left(P_{win1,win}\right)$, $n_{win1,win}=\mathrm{card}\left(P_{win2,win}\right)$

Ces formules itératives ont l'avantage d'estimer directement le modèle $\varphi\left(\bullet,\theta^t_{merg,win}\right)$ à partir uniquement des paramètres $\theta^t_{win1,win}$ et $\theta^t_{win2,win}$. Ce qui évite d'effectuer à nouveau un apprentissage du modèle de fusion avec toutes les données des prototypes fusionnés. De cette

façon, la procédure de fusion n'affecte pas la rapidité de l'algorithme AUDyC. Après la fusion, le modèle du prototype résultant est mis à jour avec la donnée X_t. Enfin, il faut noter que si les prototypes fusionnés appartiennent à la même classe, le prototype résultant de leur fusion est affecté à cette même classe. Si par contre ces prototypes sont issus de classes différentes, la fusion de ces prototypes entraînera la fusion de leurs classes respectives.

La fusion de classes est également effectuée pour des classes partageant des données ambiguës sans que ces données appartiennent à leurs prototypes (cas 3 (b) du tableau 4.1). Si le nombre des données ambiguës est supérieur au seuil d'ambiguïté N_{amb}, les classes concernées C_{win} seront fusionnées en une seule C_{merg}. La classe C_{merg} est alors constituée par l'ensemble des prototypes $P_{j,win}, j \in [J_{win}]$ contenus dans chacune des classes C_{win}. Le modèle ψ_{merg} est ainsi défini avec les fonctions de densité $\varphi\left(\bullet, \theta^t_{j,win2}\right)$ de la manière suivante :

$$\psi_{merg}(X) = \sum_{win} \sum_{j=1}^{J_{win}} \varphi\left(X, \theta^t_{j,win}\right) \qquad 4.22$$

L'expérimentation de cette nouvelle procédure de fusion est exposée dans [Amadou & al., 2005b] et dans les simulations du chapitre 5 (partie 5.3).

4.2.3.4 Procédure d'Évaluation du modèle de la partition dynamique

La procédure d'évaluation a pour rôle de traiter certains défauts de modélisation et d'éliminer les classes parasites. Ceci est effectué à travers l'évaluation de la représentativité des classes toutes les T données. Nous proposons des critères et des règles servant d'une part à la scission de classes et d'autre part à l'élimination de classes.

a) Scission de classes

Au cours de la classification dynamique, certains défauts de modélisation peuvent se traduire par de fortes discontinuités d'une classe décrite par plusieurs prototypes (figure 4.5.a). Dans ces situations, il est nécessaire de scinder cette classe et redéfinir les modèles exacts de toutes les classes constituées par les données. Nous proposons un critère de scission de classes basé sur une technique de recherche heuristique. Cette heuristique vise à retrouver tous les groupes des prototypes décrivant une région continue. Elle utilise la distance de Fisher F_{sh} pour mesurer la proximité des prototypes. F_{sh} est définie par :

$$F_{sh}\left(\theta^t_{j,m}, \theta^t_{k,m}\right) = \left(\bar{X}^t_{j,m} - \bar{X}^t_{k,m}\right)\left({\Sigma^t_{j,m}}^{-1} + {\Sigma^t_{k,m}}^{-1}\right)\left(\bar{X}^t_{j,m} - \bar{X}^t_{k,m}\right)^T \qquad 4.23$$

où $\theta^t_{j,m}$ et $\theta^t_{k,m}$ sont les paramètres de deux prototypes $P_{j,m}$, $P_{k,m}$ d'une même classe C_m.

La recherche heuristique est effectuée dans chacune des classes C_m de façon à détecter les prototypes suffisamment proches pour constituer des classes homogènes C_{split}. A l'issue de cette recherche, deux résultats sont possibles :

- Il n'y pas de scission si la classe C_m est constituée d'un seul prototype, ou bien chaque prototype $P_{j,m} \subset C_m$ est suffisamment proche d'un autre prototype $P_{g,m} \subset C_m$, i.e. :

$$\forall P_{j,m},\ \exists P_{k,m} / F_{sh}\left(\theta^t_{j,m}, \theta^t_{k,m}\right) \leq d_{fsh} \qquad 4.24$$

où d_{fsh} est le seuil de scission.

- Sinon, la classe C_m est scindée en deux ou plusieurs classes C_{split} de telle sorte que :

$$\left\{\bigcup_{split} C_{split}\right\} = C_m \ / \begin{cases} C_{split} = !P_{j,m} & \Leftrightarrow \ \forall P_{g,m} \subset C_m,\ F_{sh}\left(\theta^t_{j,m}, \theta^t_{g,m}\right) > d_{fsh} \quad (! : \text{unique}) \\ C_{split} = \left\{P_{j,m}, P_{g,m}, \ldots\right\} & \Leftrightarrow \ \forall P_{j,m} \subset C_{split},\ \exists P_{g,m} \subset C_{split} / F_{sh}\left(\theta^t_{j,m}, \theta^t_{g,m}\right) \leq d_{fsh} \end{cases} \qquad 4.25$$

Après la scission d'une classe, il reste à déterminer le modèle ψ_{split} de chacune des classes C_{split} (figure 4.5.b). L'opération de scission s'exprime simplement par la relation :

$$\psi_{split}(X) = \sum_{P_{j,m} \subset C_{split}} \varphi\left(X, \theta^t_{j,m}\right) \qquad 4.26$$

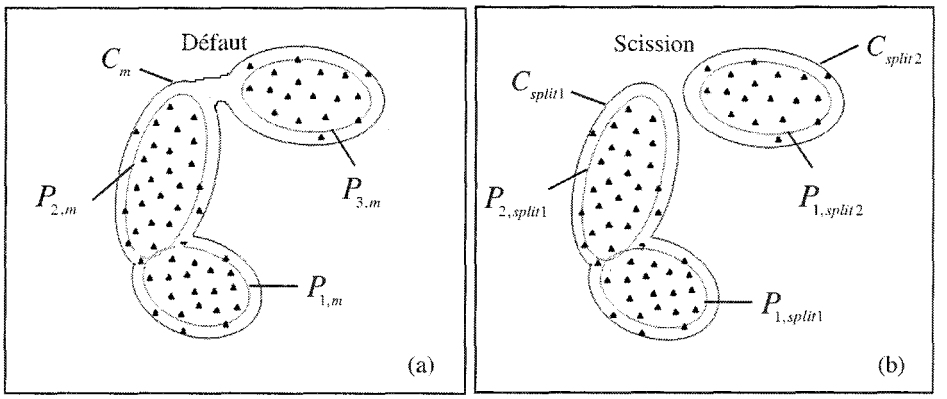

Figure 4.5 : Scission de classe. La classe $C_m = \left\{P_{1,m}, P_{2,m}, P_{3,m}\right\}$ (a) est scindée en deux classes $C_{split1} = \left\{P_{1,split1}, P_{2,split1}\right\}$ et $C_{split2} = \left\{P_{1,split2}\right\}$ (b).

Le mécanisme de scission proposé dans cette sous-section ne concerne que les classes. Suivant la même démarche, on peut envisager un mécanisme de scission de prototypes en utilisant une heuristique détectant la proximité des données dans chaque prototype. Un mécanisme similaire est proposé dans [Chaudhuri & al., 1992]. Il est évident qu'un tel mécanisme accroîtra considérablement le temps d'exécution de l'algorithme AUDyC et pourra compromettre son utilisation dans les problèmes en ligne.

b) Élimination des classes parasites

La qualité de modélisation des classes dépend de celle des prototypes. Les prototypes de faible cardinalité affectent la qualité de modélisation. Les données contenues dans ces prototypes sont insuffisantes pour construire des modèles cohérents. On utilise alors un critère de cardinalité appliqué directement aux prototypes pour détecter ceux de cardinalité faible. Le

seuil de cardinalité $N_{\min}$ sera choisi en fonction du bruit. La suppression des prototypes non-représentatifs entraîne l'élimination des classes parasites (généralement dues au bruit). Par ailleurs, les prototypes gaussiens ayant des matrices de covariance singulières (mal conditionnées) sont la cause de certains défauts de modélisation constatés au sein des classes de l'AUDyC. Ces prototypes sont également éliminés. Les données constituant les prototypes éliminés sont stockées dans un ensemble noté χ^{rej}. Ces données seront repassées dans l'apprentissage au cours des itérations futures ou à la fin de la classification.

L'élimination de classes est très discutable selon les applications. Par exemple, dans un contexte de supervision de processus industriels, l'élimination de classes (modes du processus) de la base d'apprentissage est une prise de risque élevé. En effet, même si ces modes sont appris dans un passé lointain, ou ont une faible cardinalité, ils peuvent être révélateurs de pannes moins fréquentes.

4.2.4 Architecture neuronale de l'AUDyC

Le réseau AUDyC est décrit suivant l'architecture neuronale présentée sur la Figure 4.6. Pour plus de clarté, nous apportons quelques précisions concernant principalement la couche cachée. Compte tenu de la méthode de modélisation utilisée, chaque neurone caché du réseau AUDyC correspond à un prototype et mémorise le paramètre $\theta^t_{j,m}$ de son modèle de définition. La fonction d'activation de chaque neurone caché est définie par le degré d'appartenance $\mu_{j,m}$. Les neurones cachés et leurs connexions avec la couche d'entrée mémorisent les éléments du vecteur $\Theta^t_m = \left[\theta^t_{1,m},...,\theta^t_{j,m},...,\theta^t_{J_m,m}\right]$. A chaque fois qu'une nouvelle donnée se présente, ces neurones sont mis en compétition afin de déterminer le prototype $P_{j,m} \subseteq C_m$ gagnant. Par ailleurs, les connexions entre les neurones cachés et le neurone de sortie représentant la classe C_m sont pondérées respectivement par les valeurs des possibilités $p_{j,m}$ des prototypes $P_{j,m} \subseteq C_m$. Pour les prototypes $P_{j,m'} \not\subset C_m$, les connexions entre les neurones de ces prototypes et le neurone de sortie de la classe C_m ont des poids nuls.

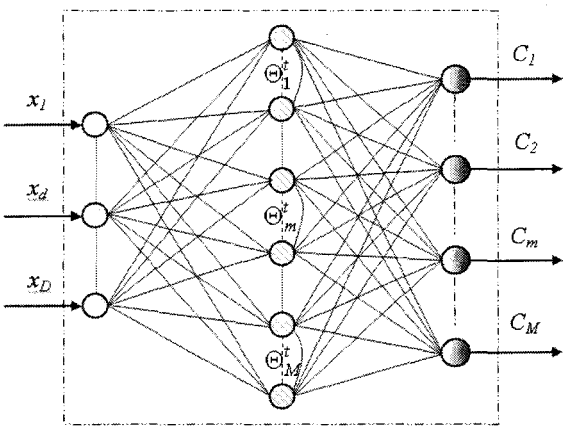

Figure 4.6 : Architecture du Réseau AUDyC

4.2.5 Performances : Convergence et Complexité

Après l'exposé de l'algorithme AUDyC, cette section présente une étude de ses performances. Dans un premier temps, la convergence de la procédure d'adaptation est étudiée en se basant sur la propriété de convergence introduite dans la description générique (chapitre 1, équation 1.19). Dans un second temps, on évalue la complexité algorithmique de l'AUDyC à travers la détermination de celle de chacune de ses procédures d'apprentissage.

4.2.5.1 Convergence de la procédure d'adaptation de l'AUDyC

L'étude de la convergence est généralement réalisée lorsque l'on connaît le résultat final que l'on souhaite obtenir. Or en classification dynamique, les cibles évoluent dans le temps, l'algorithme produit des résultats successifs dont on ne dispose d'aucune connaissance *a priori*. Afin d'analyser les propriétés de convergence des classifieurs dynamiques, nous proposons d'étudier le comportement des erreurs commises tout au long de l'apprentissage. Ces erreurs sont mesurées pour l'AUDyC comme la différence entre le modèle estimé par l'algorithme à chaque instant et le modèle qu'aurait donné l'estimateur théorique du modèle gaussien. En effet, si ces erreurs divergent l'algorithme peut avoir un comportement chaotique. La convergence de l'AUDyC dépend fortement de celle de la procédure d'adaptation. Cette procédure de mise à jour récursive est le « moteur » de l'algorithme. Elle est le facteur le plus déterminant sur la qualité du modèle de classification, si on fait l'hypothèse que les autres procédures (fusion ou scission) ne causent des défauts majeurs.

Proposition 4.1 : La procédure d'adaptation du classifieur AUDyC est convergente.

Il suffit de constater que la procédure d'adaptation de l'AUDyC est établie avec une règle de mise à jour exacte. En effet, cette règle n'est que la mise en forme récurrente de l'estimateur théorique de densité gaussienne (Annexe A.3). On en déduit donc que l'erreur entre le modèle estimé par la procédure d'adaptation récursive et le modèle gaussien théorique converge vers 0 lorsqu'il y a suffisamment de données.

Afin d'illustrer la convergence de la procédure d'adaptation de l'AUDyC, on utilise la distance de Kullback-Leibler comme mesure d'erreur. En effet, cette distance constitue une mesure d'erreur adéquate prenant en compte les erreurs de position et les dissimilarités de forme et de volume entre deux modèles gaussiens. En supposant θ_{th} le paramètre du prototype gaussien théorique P_{th} calculé sur l'échantillon $\chi_{ech} = \{X_1, ..., X_L\}$ et θ^t_{update} le paramètre du prototype gaussien P^t_{update} estimé à chaque instant t par la procédure d'adaptation récursive, l'erreur entre ces modèles est définie par l'expression suivante :

$$\xi(\theta^t_{update}, \theta_{th}) = tr\left((\Sigma_{th}^{-1} + \Sigma^t_{update}{}^{-1})(\bar{X}_{th} - \bar{X}^t_{update})(\bar{X}_{th} - \bar{X}^t_{update})^T\right) + tr\left(\Sigma_{th}\Sigma^t_{update}{}^{-1} + \Sigma^t_{update}\Sigma_{th}^{-1}\right) - 2.D \quad 4.27$$

Les résultats de la figure 4.7 illustrent la convergence de la procédure d'adaptation de l'AUDyC pour différentes valeurs de Σ_{ini}. 400 données gaussiennes sont utilisées. On constate que la rapidité de convergence de l'algorithme dépend du réglage de la matrice de covariance

initiale. On peut également montrer la convergence de cette procédure d'adaptation pour des classes complexes (multi-prototypes).

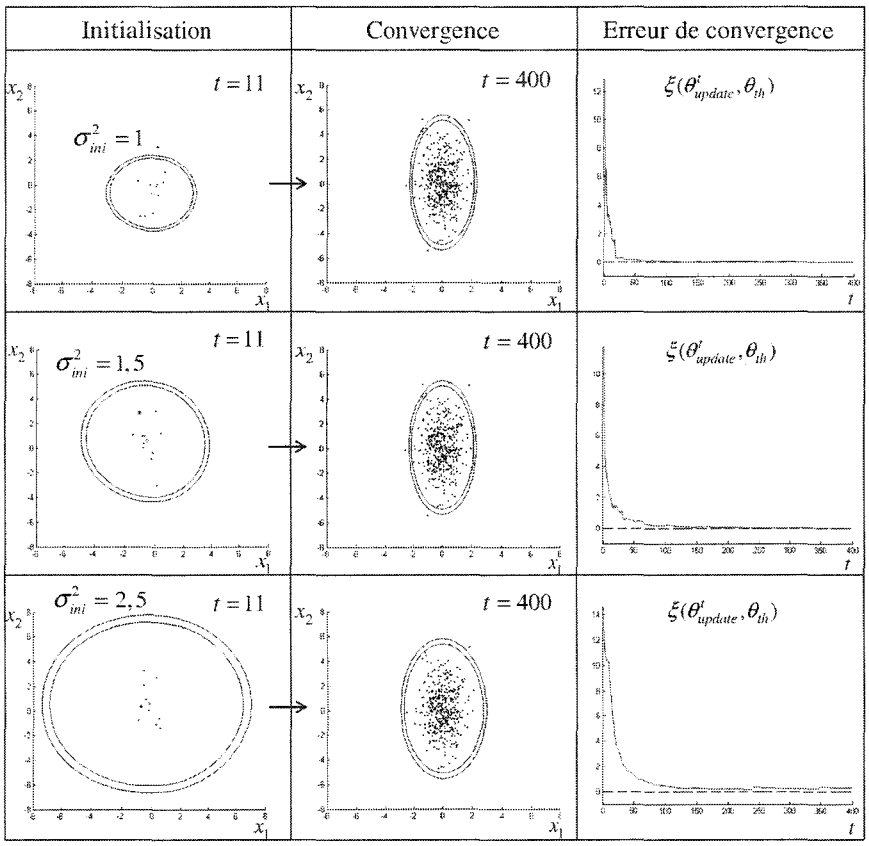

Figure 4.7 : Convergence de la procédure d'adaptation de l'AUDyC sur un prototype gaussien

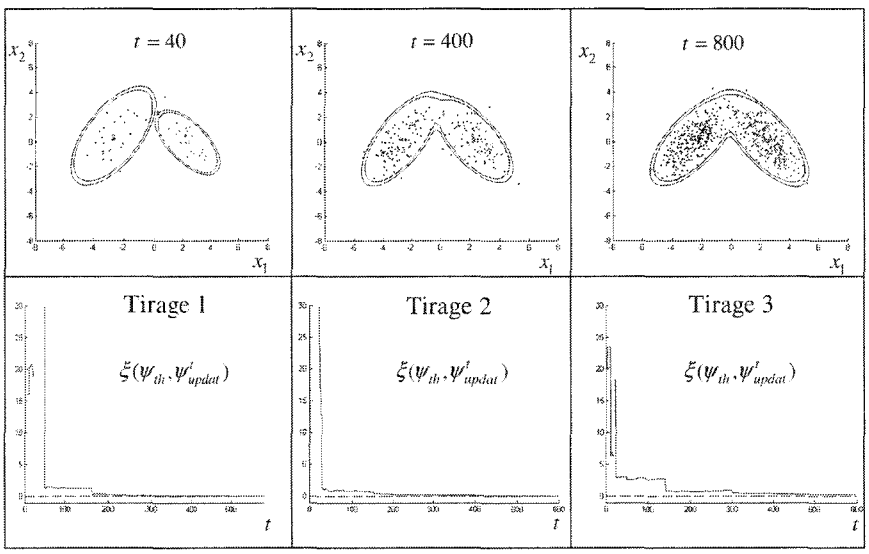

Figure 4.8 : Convergence de la procédure d'adaptation récursive de l'AUDyC sur une classe complexe constituée de deux prototypes gaussiens.

En supposons ψ_{th} le modèle théorique d'une classe complexe et ψ^t_{update} son estimation temporelle à partir de la procédure d'adaptation récursive, l'erreur de convergence est donnée

par la moyenne des erreurs de convergence de tous les prototypes de cette classe. La figure 4.8 donne illustre la convergence de la procédure d'adaptation en utilisant une classe complexe bi-prototypes. Cette expérience est réalisée sur plusieurs tirages en changeant, de façon aléatoire, l'ordre chronologique des données. Les courbes de convergence de trois tirages montrent la convergence de l'algorithme (figure 4.8).

4.2.5.2 Complexité de l'AUDyC

Dans le cadre de la classification dynamique, nous recherchons non seulement une meilleure qualité de modélisation mais aussi une exécution à moindre coût de calcul. Dans cette section, la complexité de l'AUDyC est évaluée en calculant le coût d'exécution de chacune de ses procédures d'apprentissage (figure 4.1). Après calcul, on constate que la complexité dépend de trois grandeurs : la dimension D de données, le nombre J de prototypes et le nombre L des données à traiter. Le calcul de la complexité[1] de différentes procédures d'apprentissage est présenté sur le tableau 4.2. Du tableau, on conclut que la complexité de l'algorithme est de l'ordre de $O(D^3 \times J^2 \times L)$, i.e. une complexité polynomiale d'ordre 3 en fonction de la dimension, d'ordre 2 en fonction du nombre de prototypes et d'ordre 1 en fonction du nombre de données. D'après le processus d'apprentissage de l'AUDyC, seul le critère de similarité est appelé à chaque acquisition (figure 4.1). Les autres procédures ne sont pas toutes utilisées de façon systématique.

Procédures d'apprentissage	Nb d'opérations	Complexité	Commentaires
Critère de similarité	$(2D^3+3D^2+2D)\times J\times L$	$O(D^3\times J\times L)$	Complexité polynomiale
Procédure Création	-	-	Affectations
Procédure d'Adaptation Ajout / Ajout et Retrait	$(D^2+2D+3)\times L$	$O(D^2\times L)$	Complexité polynomiale
	$(3D^2+5D+3)\times L$	$O(D^2\times L)$	
Procédure de Fusion Critère / Opération	$8D^3-2D^2+6D-1$	$O(D^3)$	Complexité polynomiale
	D^2+4D+4	$O(D^2)$	
Procédure de Fusion	$(4D^3-5D^2+D-1)\times\sum_{m=1}^{M}\frac{(J_m-1)J_m}{2}$	$O(D^3\times J^2)$	Complexité polynomiale
Procédure d'Élimination	-	-	Suppressions

Tableau 4.2 : Calcul de la complexité des procédures d'apprentissage de l'AUDyC

4.2.6 Discussion sur les paramètres de l'AUDyC

La technique de modélisation est une approche paramétrique qui n'est pas à l'abri des problèmes d'optima locaux. Ces défauts de modélisation sont souvent causés par un mauvais

[1] Le nombre d'opérations pour une inversion de matrice est calculé en se basant sur la méthode de Gauss-Jordan appelée également méthode de Pivot [Nougier, 1993].

réglage des paramètres de l'algorithme. En effet, l'AUDyC est critiquable pour son nombre important de paramètres. Nous classons ces paramètres en trois groupes :

- *les paramètres du modèle* : le paramètre σ_{ini} initialisant la matrice de covariance des prototypes, les seuils $\mu_{\min}$ et $\mu_{\max}$ d'appartenance respectifs aux prototypes et aux classes et la taille N_P de la fenêtre de définition du prototype gaussien.
- *les seuils de fusion et de scission* : le seuil de discordance ε_{kl} et le seuil de scission d_{fsh}. On fixe $\varepsilon_{kl} = 3,2$ et $d_{fsh} = 9$ par expérience.
- *Les paramètres de robustesse* : le seuil d'ambiguïté N_{amb}, le seuil de cardinalité $N_{\min}$, et le nombre d'échantillons T entre deux procédures d'évaluation.

On peut remarquer que la plupart des paramètres sont utilisés pour conférer de la flexibilité à l'algorithme et accroître son champ d'application. Nous proposons en Annexe A.3 une méthode de réglage des paramètres σ_{ini}, $\mu_{\min}$, $\mu_{\max}$ dont le choix pose quelques difficultés.

4.3 Self-Adaptive Kernel Machine (SAKM)

Développé à partir des méthodes à noyau, l'algorithme SAKM est récemment introduit pour la classification dynamique [Amadou & Lecoeuche, 2005a]. C'est une contribution originale pour les SVM & méthodes à noyau dans les problèmes d'apprentissage en ligne. Comme la nouvelle version de l'AUDyC, l'algorithme SAKM est construit suivant les principes donnés dans la description générique (chapitre 1, partie 1.4). Les classes du SAKM sont modélisées à l'aide de fonctions d'apprentissage à noyau représentées par des hyperplans ou supports de distribution dans l'espace RKHS *(Reproduising Kernel Hilbert Space)*. Grâce à ses procédures d'apprentissage, l'algorithme dispose de capacités d'auto-adaptation en environnement de données non-stationnaires. Le critère de similarité du SAKM est basé sur une métrique élaborée dans l'espace RKHS. C'est un critère de similarité exploitant la propriété du noyau reproduisant. De même, nous proposons une nouvelle règle de mise à jour permettant l'adaptation itérative des supports de distribution dans l'espace RKHS. Basée sur la technique du gradient, cette procédure est très rapide. Elle est conçue pour prendre en compte les modifications locales et les évolutions avec glissements de classes.

Nous allons tout d'abord décrire la technique de modélisation du SAKM. Ensuite, nous présentons son critère de similarité à noyau défini dans l'espace RKHS. Par la suite, les procédures d'apprentissage du SAKM sont détaillées. Enfin, après avoir expliqué l'architecture neuronale du réseau SAKM, la section 4.3.5 est consacrée à l'étude de la convergence et de la complexité de l'algorithme. Cette partie est clôturée par une discussion.

4.3.1 Modèle de classification du SAKM

Le modèle de classification du SAKM est décrit à l'aide des **fonctions d'apprentissage à noyau**. Celles-ci sont des modèles non-paramétriques capables de modéliser des classes complexes. Dans l'espace RKHS, chacune de ces fonctions est représentée par un hyperplan appelé **support de distribution** (ou encore ensemble de niveau). Le classifieur mono-classe (One-Class-SVM) détaillé dans le chapitre 2 (section 2.3.2) s'est montré très performant lorsqu'il s'agit d'estimer le modèle d'une classe dont toutes les données sont connues *a priori*. Cependant, la classification dynamique est un problème séquentiel, multi-classe et non-supervisé. Nous proposons alors de décrire le modèle de classification du SAKM en représentant chaque classe C_m de la partition dynamique Π par un support de distribution dans l'espace RKHS. En supposant ψ_m la fonction d'apprentissage à noyau modélisant la classe C_m dans l'espace χ, ψ_m est définie à chaque instant t de la manière suivante :

$$\psi_m^t(X) = \sum_{j=1}^{J_m} \alpha_{j,m}^t \kappa\left(X, SV_{j,m}^t\right) - \rho_m^t \qquad 4.28$$

où $SV_{j,m}^t$ et $\alpha_{j,m}^t$ sont respectivement le $j^{ème}$ vecteur support et son poids dans la définition de la fonction d'apprentissage ψ_m^t. Le paramètre ρ_m^t est l'offset de ψ_m^t. Par analogie avec la

modélisation de l'AUDyC, le couple $\left(SV_{j,m}^t, \alpha_{j,m}^t\right)$ correspond au paramètre $\theta_{j,m}^t$ du prototype gaussien $P_{j,m}$. ψ_m^t a pour paramètre le vecteur $\Theta_m^t = \left[\left(SV_{1,m}^t, \alpha_{1,m}^t\right), \ldots, \left(SV_{j,m}^t, \alpha_{j,m}^t\right), \ldots, \left(SV_{J_m,m}^t, \alpha_{J_m,m}^t\right)\right]$.

$\kappa(\cdot,\cdot)$ correspond au noyau gaussien choisi pour ses bonnes propriétés montrées dans le chapitre 2, section 2.3.2.5. Rappelons la définition de ce noyau :

$$\kappa(X_1, X_2) = \exp\left(-\lambda \|X_1 - X_2\|\right), \quad \forall X_1, X_2 \in \chi \qquad 4.29$$

où λ représente le paramètre du noyau gaussien.

Suivant cette modélisation, chaque classe du SAKM est définie par l'ensemble :

$$C_m = \{X \in \chi / \psi_m(X) \geq 0\} \qquad 4.30$$

La frontière de la classe C_m est déterminée par l'équation de l'hyperplan : $\Delta_m : \psi_m(X) = 0$. La figure 4.9 illustre la modélisation de 3 classes du SAKM dans les espaces χ et Γ.

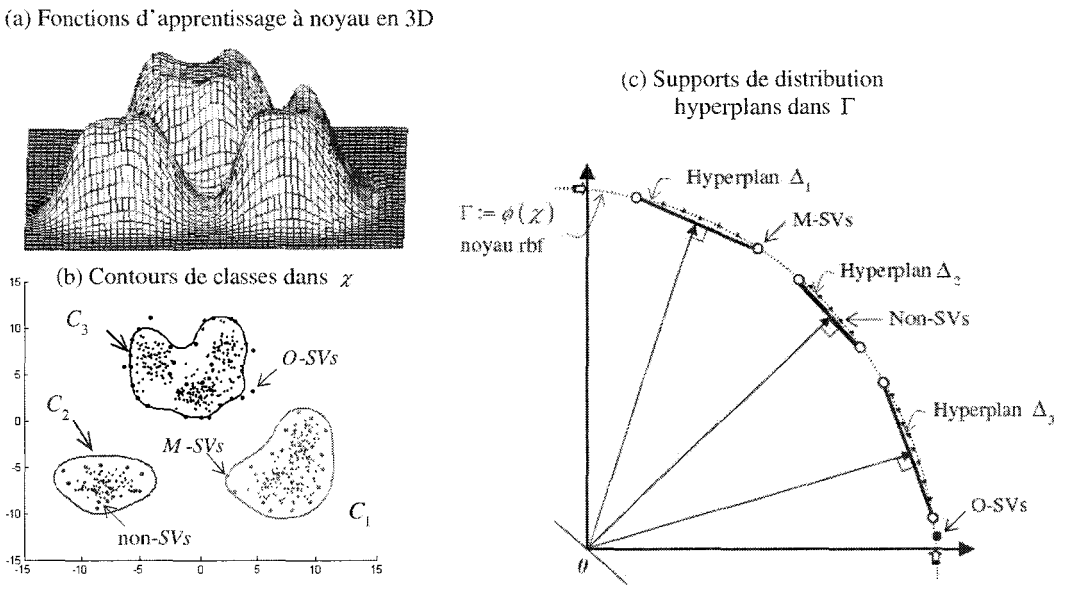

Figure 4.9 : Modèle de classification du SAKM. Les fonctions d'apprentissage à noyau sont représentées par des hyperplans (supports de distribution) de classes dans RKHS.

Afin de prendre en compte les évolutions des classes, leurs supports de distribution sont adaptés en utilisant les procédures d'apprentissage développées dans la section 4.3.2.

Fonction objectif

Pour mesurer les performances du SAKM en terme d'erreurs de classification, nous définissons une fonction objectif notée E_{learn}. Cette fonction mesure la moyenne des erreurs cumulées au cours de l'apprentissage à chaque instant t. Elle est établie de la façon suivante :

$$E_{learn}\left[\Im^{(t)}\right]=\frac{1}{M}\sum_{m=1}^{M}\frac{1}{\mathrm{card}(C_m)}\sum_{i=1}^{t}\xi\left(\psi_m^t, X_i\in C_m\right) \qquad 4.31$$

où M est le nombre de classes et ξ correspond à l'erreur de « Hinge » définie par :

$$\xi(\psi, X)=\max\left(0,-\psi(X)\right)-\nu\rho \qquad 4.32$$

où ν est un coefficient pris dans l'intervalle [0,1].

La fonction objectif, telle qu'elle est définie dans l'équation (4.31), est une mesure adéquate de la performance du classifieur dynamique. En effet, elle quantifie l'importance des données classées avec erreurs (i.e. *outliers*). La minimisation de cette fonction objectif par l'algorithme SAKM conduit à une bonne qualité de classification.

Nous venons de décrire le modèle de classification du SAKM et nous avons défini une fonction objectif pour évaluer ses performances. Avant de détailler les procédures d'apprentissage de l'algorithme du SAKM, définissons tout d'abord son critère de similarité.

4.3.2 Critère de similarité à noyau dans RKHS

Dans les méthodes SVM, la fonction d'apprentissage est généralement utilisée comme la mesure de similarité entre les données et les classes. Par exemple, pour la classification SVM mono-classe, elle sert de fonction de décision. En effet, cette fonction prend des valeurs positives pour les données situées à l'intérieur du contour de classe, et à l'opposé, pour les données situées à l'extérieur de ce contour, elle prend des valeurs négatives. Cependant, les bornes de cette fonction ne sont pas connues. Une idée intuitive consiste à établir une mesure de similarité en utilisant les fonctions d'apprentissage des classes définies indépendamment les unes des autres. Cette solution conduit à des erreurs dans un espace multi-classe. Une autre idée proposée dans [Desobry & al., 2005] consiste à mesurer dans l'espace RKHS, l'angle entre la nouvelle donnée et le point central de l'hyperplan de chaque classe. Cette technique ne peut être utilisée que si le noyau choisi vérifie certaines propriétés. Il s'agit plus précisément des conditions imposées sur les bornes du noyau [Desobry & al., 2005].

Nous proposons d'introduire une nouvelle mesure de similarité à noyau pour le SAKM. Le principe est simple. Il consiste à mesurer la distance entre chaque nouvelle donnée X_t et le vecteur support le plus proche pour chaque classe C_m dans l'espace RKHS. L'espace RKHS étant doté du produit scalaire, il est alors possible de calculer cette distance en utilisant une norme [Borer, 2003]. La nouvelle donnée X_t est correctement classée dans la classe C_m si elle est acquise à l'intérieur du contour ou dans l'ensemble de support vecteurs de la marge. Sinon, si elle est dans l'ensemble des *outliers* de C_m, elle est affectée à cette classe avec une erreur de tolérance (*slack* error). La mise en équation de cette idée est facilitée par la représentation géométrique de l'espace Γ (figure 4.10).

Tout d'abord, une compétition entre les vecteurs support $SV^t_{j,m}$ de chaque classe C_m permet de trouver le vainqueur $SV^t_{ppv,m}$ c'est-à-dire le plus proche de la donnée X_t :

$$ppv = \arg\min_j \left\| \phi(X_t) - \phi\left(SV^t_{j,m}\right) \right\| \qquad 4.33$$

Ensuite, nous définissons la mesure de similarité $S(X_t, C_m) = \mu\phi_{t,m}$ de la manière suivante :

$$\mu\phi_{t,m} = 1 - \frac{\delta}{\sqrt{2}} \left\| \phi(X_t) - \phi\left(SV^t_{ppv,m}\right) \right\| = 1 - \delta \times \sqrt{1 - \kappa\left(X_t, SV^t_{ppv,m}\right)} \qquad 4.34$$

$$\text{où } \delta = \begin{cases} 1 & \text{si } \psi^t_m(X_t) < 0 \\ 0 & \text{sinon} \end{cases}$$

Enfin, en utilisant le noyau gaussien, l'expression (4.34) devient :

$$\mu\phi_{t,m} = 1 - \delta \times \sqrt{1 - \exp\left(-\lambda \left\| X_t - SV_{win,m} \right\|\right)} \qquad 4.35$$

La mesure de similarité $\mu\phi_{t,m}$ telle que définie, est bornée dans l'intervalle [0,1] et est strictement monotone par rapport à la variable $\left\| X_t - SV_{win,m} \right\|$. On a les relations suivantes :

$$\begin{aligned} \left\| X_t - SV_{win,m} \right\| = 0 &\Leftrightarrow S(X_t, C_m) = 1 \\ \left\| X_t - SV_{win,m} \right\| \to \infty &\Leftrightarrow S(X_t, C_m) \to 0 \end{aligned} \qquad 4.36$$

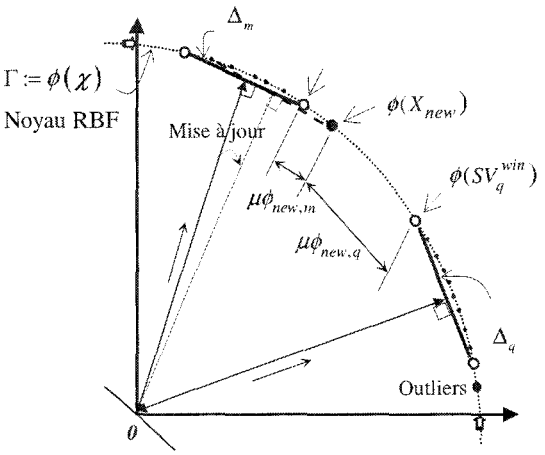

Figure 4.10 : Mesure de similarité entre une nouvelle donnée X_{new} et deux classes C_m (hyperplan de support Δ_m) et C_q (hyperplan de support Δ_q) dans l'espace de Hilbert à noyau reproduisant Γ.
Les données sont transformées de l'espace χ vers l'espace Γ en utilisant un noyau RBF.

Pour trouver l'ensemble Π^{win} de classes gagnantes à chaque instant t, le critère de similarité du SAKM est définie de la manière suivante :

$$\Pi^{win} = \{C_m / \mu\phi_{t,m} \leq \varepsilon_{th}\} \qquad 4.37$$

où ε_{th} est le seuil de similarité. Ce seuil est fixé à $\varepsilon_{th} = 0,2$. Ce choix est effectué de façon à respecter la contrainte de continuité de la région décrite par chaque classe dans χ.

Motivation : Supposons que la donnée X_t soit proche de la classe C_m. L'affectation de X_t à la classe C_m ne doit se faire que si cette opération n'affecte pas la continuité de la région décrite par cette classe. Considérant la représentation de C_m dans l'espace χ (figure 4.11) et la géométrie du noyau RBF, la contrainte de continuité impose la condition suivante :

$$X_t \in C_m \text{ si } \left\| X_t - SV_{win.m} \right\| \leq 2.\sigma = \frac{1}{\lambda}$$

Sous cette condition, on a :

$$\mu\phi_{t,m} \geq 1 - \sqrt{1 - \exp(-1)} = 1 - 0,79 \approx 0,2$$

Nous choisissons ainsi de fixer le seuil ε_{th} à $0,2$.

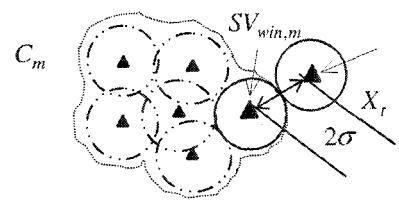

Figure 4.11: Dans l'espace d'entrée χ, la distance maximale entre les données de chaque classe doit être inférieure au double du rayon du noyau RBF pour conserver la continuité de la classe.

4.3.3 Processus d'apprentissage du SAKM

Le processus d'apprentissage de l'algorithme SAKM est constitué des 4 procédures proposées dans la description générique du classifieur dynamique (chapitre 1, partie 1.4). La règle de décision est définie à partir du critère (4.37) et donnée dans le tableau 4.3.

Cas 1	card(Π^{win}) = 0	Procédure de création
Cas 2	card(Π^{win}) = 1	Procédure d'adaptation
Cas 3	card(Π^{win}) ≥ 2	Procédure de fusion
Cas 4	Toutes les T données	Procédure d'évaluation

Tableau 4.3 : Règle de décision et procédures d'apprentissage du SAKM

4.3.3.1 Procédure de Création : Initialisation de modèle, Création de classes

Au départ, on fait l'hypothèse que la fonction de pré-initialisation est $\psi_0 = 0$ (modèle de la classe « vide » par convention). A l'acquisition de la première donnée X_1 donnant lieu à la création d'une classe C_1, l'initialisation du modèle ψ_1 est défini avec les paramètres :

$$\Theta_1 = \left(\alpha_{new.new}^{t=1} = 1, \quad SV_{1,1}^{t=1} = X_t \right) \text{ et } \rho_1^{t=1} = \eta \qquad 4.38$$

où η est le ratio d'apprentissage (section 4.3.3.2).

Ensuite, pour chaque nouvelle donnée X_t donnant lieu à la création d'une nouvelle classe C_{new} (Cas 1, tableau 4.3), le mécanisme de création est le même. Un modèle ψ_{new} est initialisé avec les paramètres :

$$\Theta_{new} = \left(\alpha^t_{new,new} = 1,\ SV^t_{1,new} = X_t\right) \text{ et } \rho^t_{new} = \eta \qquad 4.39$$

À l'instant de sa création, la classe C_{new} ne contient que la donnée X_t (détectée comme un vecteur support car $\psi_0(X_t) = 0$). Cette donnée correspond donc au seul vecteur support $SV^t_{1,new}$ de la classe C_{new} à l'instant de sa création. Rappelons que la création de classe entraîne l'incrémentation des ensembles $\Im^{(t)}$ et $\Pi^{(t)}$.

4.3.3.2 Procédure d'Adaptation : Mise à jour de support de distribution

Dans le cas 2 (tableau 4.3), la donnée X_t est suffisamment proche d'une seule classe C_{win} pour participer à sa définition. En s'inspirant du NORMA (chapitre 3, section 3.2.2.3), nous proposons une procédure d'adaptation itérative basée sur la technique du gradient. L'objectif recherché à travers cette procédure est l'adaptation du modèle face aux modifications locales et aux évolutions avec glissements. L'élaboration de la procédure d'adaptation du SAKM est effectuée en trois étapes :

- Estimation des paramètres $\alpha^t_{j,win}$ du modèle ψ_{win} par le gradient stochastique en minimisant le Risque Instantané dans l'espace RKHS.
- Normalisation des paramètres $\alpha^t_{j,win}$ afin de respecter la contrainte introduite pour l'estimateur théorique One-Class-SVM (chapitre 2, équation 2.45).
- Détermination de l'offset ρ_{win} en utilisant l'équation de l'hyperplan Δ_{win} dans l'espace RKHS de telle sorte que :

$$\Delta_{win} : \psi_{win}\left(SV^t_{c,win}\right) = 0 \ \Rightarrow\ \rho^t_{win} = \sum_{j=1}^{J_m} \alpha^t_{j,win} \kappa\left(SV_{c,win}, SV^t_{j,win}\right),\ c \in [1,\ldots,J_m] \qquad 4.40$$

La première étape d'estimation des paramètres $\alpha^t_{j,win}$ est expliquée dans la présentation du NORMA (chapitre 3, section 3.2.2.3). Après la normalisation des $\alpha^t_{j,win}$ et la détermination de ρ_{win}, on obtient les formules de mise à jour itératives des paramètres de ψ_{win} :

$$\left\{\begin{array}{l} \left\{\begin{array}{ll} \alpha^t_{j,win} = (1-\eta)\alpha^{t-1}_{j,win} & \text{si } t-\tau < j < t \\ \alpha^t_{new,win} = \eta\ [\text{resp. } 0] & \text{si } \psi^{t-1}_m(X_t) < 0\ [\text{resp.} \geq 0] \end{array}\right\}, \text{ puis } \alpha^t_{j,win} = \alpha^t_{j,win} \Big/ \sum_{j=1}^{J_m} \alpha^t_{j,win} \\ \rho^t_{win} = \sum_{j=1}^{J_{win}} \alpha^t_{j,win} \kappa\left(SV_{c,win}, SV^t_{j,win}\right),\quad c \in [J_{win}] \end{array}\right. \qquad 4.41$$

où η est le ratio d'apprentissage. Il est choisi constant dans l'intervalle $[0,1]$.

Avec cette procédure d'adaptation, l'apprentissage de la classe C_{win} est effectué sur une fenêtre exponentielle de taille $J_{win} = \max\left(\text{card}(C_{win}), \tau\right)$, où τ représente le nombre maximal de vecteurs support de la classe. L'intérêt de ce paramètre est de permettre l'évolution de la classe modèle en élaguant les informations obsolètes. Par la même occasion, le nombre de calcul est réduit en limitant l'expansion du modèle. On remarque que lorsque la donnée X_t est acquise à l'intérieur du contour de la classe C_{win} (i.e. $\psi_m^{t-1}(X_t) \geq 0$), aucun calcul n'est effectué. Dans le cas contraire, le contour de cette classe est adapté en ajoutant un nouveau vecteur support $SV_{new,win}^t = X_t$ de poids $\alpha_{new,win}^t$ (non nul). Si $J_m = \tau$, le plus ancien vecteur support $SV_{old,win}^t$ est exclu de la fenêtre d'apprentissage.

La procédure d'adaptation du SAKM a la capacité d'effectuer le suivi de classes évolutives. Cette procédure est très rapide mais elle produit des modèles approximatifs. Dans la section 4.3.5, nous démontrerons la convergence théorique de la procédure d'adaptation du SAKM avant d'évaluer la complexité de l'algorithme.

4.3.3.3 Procédure de Fusion : Fusion des classes

La procédure de fusion du SAKM est développée pour traiter les données ambigües (cas 3, tableau 4.3). Lorsque le nombre des données ambigües c'est-à-dire les données partagées par deux ou plusieurs classes dépasse le seuil d'ambiguïté N_{amb}, ces classes seront fusionnées. En revanche, contrairement à la procédure de fusion dédiée aux prototypes gaussiens (section 4.2.3.3), ici, on n'a pas besoin d'un second critère visant à analyser les modèles candidats à la fusion. En effet, le SAKM utilise des fonctions non-paramétriques capables de modéliser les classes de fusion avec de grandes capacités de généralisation. Il est néanmoins nécessaire de disposer au préalable d'une règle de fusion capable d'estimer convenablement le modèle des classes résultantes de la fusion. En posant C_{merg} le modèle d'une classe résultant de la fusion de deux ou plusieurs classes C_{win}, nous proposons deux solutions pour l'estimation de C_{merg} :

- La première solution consiste à déterminer le modèle ψ_{merg} en le réinitialisant et en appliquant la procédure d'adaptation du SAKM sur l'ensemble des données des classes C_{win} :

$$C_{merg} = \left\{X \in \left(\cup C_{win}\right),\ \psi_{merg}(X) \geq 0\right\} \qquad 4.42$$

Cette solution non adoptée pour l'AUDyC, ne pose pas de problème de complexité pour le SAKM car sa procédure d'adaptation est très rapide (section 4.3.5.2).

- La seconde solution envisageable consiste à approximer le modèle ψ_{merg} avec la somme des modèles ψ_{win} (sans leurs offsets respectifs) des classes à fusionner :

$$\psi_{merg}(\bullet) = \sum_{win}\left(\sum_{j=1}^{J_{win}} \alpha_{j,win}^t \kappa\left(\bullet, SV_{j,win}\right)\right) - \rho_{merg} \qquad 4.43$$

L'offset ρ_{merg} est calculé en utilisant l'équation : $\psi_{merg}(X)=0$. Cependant, en appliquant la seconde solution, le modèle ψ_{merg} n'est pas décrit suivant une fenêtre exponentielle. On peut approcher la fenêtre exponentielle en réarrangeant les paramètres $\alpha_{j,merg}$ suivant un ordre décroissant. De cette façon, au cours de l'adaptation récursive de la classe C_{merg}, les vecteurs support $SV_{j,merg}$ de poids faible sont oubliés au fur et à mesure.

Pour l'algorithme SAKM, nous adoptons la première règle de fusion pour sa simplicité de mise en œuvre et la structure du modèle qu'il offre.

4.3.3.4 Procédure d'Évaluation de la représentativité des classes

Afin d'évaluer la représentativité des classes, deux mécanismes sont proposés dans la description générique de classifieurs. Il s'agit d'une part, de la scission des classes qui présentent de fortes discontinuités dans leur région de définition et d'autre part, l'élimination de classes parasites et obsolètes.

La scission de classes est une procédure difficile à mettre en œuvre pour le SAKM. La difficulté principale réside dans l'élaboration d'un critère permettant de détecter les fortes discontinuités dans les classes. Un tel critère peut être établi en utilisant une heuristique qui recherche dans chaque classe, les données suffisamment proches pour former des groupes homogènes. Cependant, une telle opération serait très complexe en terme de coût de calcul et non utilisable dans les applications en ligne. Afin de ne pas trop accroître la complexité du SAKM, cet algorithme n'est pas doté de procédure de scission. Par ailleurs, l'estimation récursive des modèles est effectuée en minimisant un risque régularisé. Or, Vapnik [1995] montre que si une méthode d'estimation de modèle conduit à la minimisation d'un tel risque, alors elle évite les problèmes d'optima locaux (défauts de modélisation).

En revanche, l'algorithme SAKM est doté d'une procédure d'élimination développée pour la suppression de classes parasites. Cette procédure est basée sur le même principe que celui proposé dans la description générique. De façon périodique, les classes de cardinalité inférieure au seuil $N_{\min}$ sont éliminées. Le choix de ce seuil dépend du bruit dans les données.

4.3.4 Architecture neuronale du SAKM

Le réseau SAKM est construit avec une architecture *feedfoward* à trois couches : une couche d'entrée, une couche cachée, une couche de sortie. Chaque neurone caché mémorise un vecteur support $SV_{j,m}$. Une compétition entre les neurones cachés permet d'activer le vecteur support vainqueur $SV_{win,m}$. Les connexions entre la couche cachée et la couche de sortie sont pondérées par les paramètres $\alpha_{j,m}$ représentant respectivement les poids des vecteurs support $SV_{j,m}$ dans le modèle ψ_m. Les poids des autres connexions sont nuls, c'est-à-dire ceux des connexions entre les neurones cachés d'indice différent de m et le neurone de sortie représentant la classe C_m. Pour compléter l'architecture du SAKM, un nœud *Bias* est

connecté à la couche de sortie. Ce nœud sert à mémoriser l'offset ρ_m de chaque classe à travers *ses connexions avec la couche de sortie. Le réseau SAKM est représenté sur la figure 4.12.

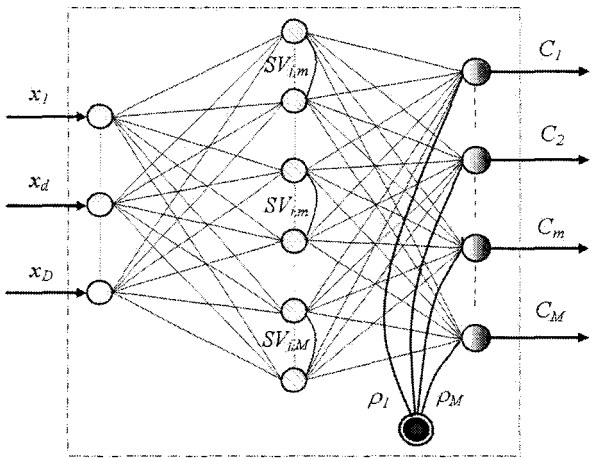

Figure 4.12 : Architecture neuronale du SAKM

Cette architecture est complètement évolutive (constructive et destructive). Elle permet la création et la suppression de neurones correspondant aux vecteurs supports et aux classes. La prise en compte de modifications locales et d'évolutions de classes est obtenue à travers l'adaptation des poids des neurones et des connexions.

4.3.5 Etude des performances du SAKM

Cette section est consacrée à l'étude des performances du SAKM. Dans un premier temps, nous allons étudier la convergence de la procédure d'adaptation de l'algorithme. Ensuite, la complexité du SAKM est évaluée en déterminant le nombre d'opérations de chacune de ces procédures d'apprentissage.

4.3.5.1 Convergence de la procédure d'adaptation du SAKM

Afin d'étudier la convergence de la procédure d'adaptation du SAKM, une analyse théorique de l'erreur de convergence est menée. Plus précisément, on étudie le comportement de la fonction objectif E_{learn} lorsque le temps devient suffisamment grand. E_{learn} étant la moyenne des erreurs cumulées (ou erreur de classification dynamique), la propriété de convergence énoncée dans la description générique (chapitre 1, équation 1.19) se vérifie de façon équivalente avec la convergence de la fonction E_{learn} . Cette analyse théorique est effectuée, dans un premier temps, pour l'adaptation récursive d'une classe. Ensuite, le résultat est généralisé au problème multi-classe.

Nous allons d'abord déterminer les bornes de la fonction d'apprentissage à noyau, ensuite celles de l'erreur E_{learn} localement estimée sur une donnée. A partir de ces résultats, nous montrons que E_{learn} tend vers une limite ℓ unique et bornée. Après cette analyse

théorique, on compare la procédure d'adaptation du SAKM avec le NORMA (*Naïve Online Reg. Minisation Algorithm*) [Kivinen & al., 2004] et le Gentile' ALMA (*Approximation Large Margin Algorithm*) [Gentile, 2001]. Cette comparaison, effectuée sur un problème de suivi d'une classe évolutive, montre les bonnes propriétés de convergence du SAKM.

a) Détermination des bornes de la fonction d'apprentissage et des bornes de l'erreur locale

Considérons ψ la fonction d'apprentissage à noyau modélisant le contour d'une classe quelconque de la partition dynamique. Montrons que cette fonction estimée par la procédure d'adaptation du SAKM ne diverge pas et calculons ses bornes.

$$\psi(\bullet) = \sum_j \alpha_j . \kappa(\bullet, SV_j) - \rho = \sum_j \alpha_j . \kappa(\bullet, SV_j) - \sum_j \alpha_j . \kappa(SV_c, SV_j) \qquad 4.44$$

En utilisant la contrainte de normalisation (équation 4.41) imposant $\sum_j \alpha_j = 1$, on a :

$$\psi(\bullet) \le \left(\sum_j \alpha_j\right).\left(\max_j \kappa(\bullet, SV_j)\right) - \left(\sum_j \alpha_j\right).\left(\min_j \kappa(SV_c, SV_j)\right)$$

$$\psi(\bullet) \le 1 \;.\; \mathrm{Up}_{\ker} - 1 \;.\; \mathrm{Lo}_{\ker}$$

De la même manière,

$$\psi(\bullet) \ge \left(\sum_j \alpha_j\right).\left(\min_j \kappa(\bullet, SV_j)\right) - \left(\sum_j \alpha_j\right).\left(\max_j \kappa(SV_c, SV_j)\right)$$

$$\psi(\bullet) \ge 1 \;.\; \mathrm{Lo}_{\ker} - 1 \;.\; \mathrm{Up}_{\ker}$$

où $\mathrm{Up}_{\ker}$ et $\mathrm{Lo}_{\ker}$ sont respectivement la borne supérieure et la borne inférieure du noyau $\kappa(\bullet,\bullet)$. $\mathrm{Up}_{\ker} \ge 0$ et $\mathrm{Lo}_{\ker} \ge 0$ sont des conditions imposées par la définition du produit scalaire dans l'espace RKHS. En effet, pour tout noyau reproduisant, on a : $\kappa(\bullet,\bullet) = <\phi(\bullet), \phi(\bullet)>_\Gamma \ge 0$.

Par conséquent, les bornes de la fonction d'apprentissage à noyau sont :

$$\mathrm{Lo}_{\ker} \le \kappa(\bullet,\bullet) \le \mathrm{Up}_{\ker} \;\Rightarrow\; \mathrm{Lo}_{\ker} - \mathrm{Up}_{\ker} \le \psi(\bullet) \le \mathrm{Up}_{\ker} - \mathrm{Lo}_{\ker} \qquad 4.45$$

Par exemple, dans le cas du noyau gaussien, on a : $0 \le \kappa(\bullet,\bullet) \le 1 \;\Rightarrow\; -1 \le \psi(\bullet) \le 1$

Maintenant, considérons $\xi(\psi, X)$ l'erreur de classification localement estimée sur la donnée X. On détermine les bornes de cette erreur de la façon suivante :

$$\xi(\psi, X) = \max(0, -\psi) - \nu\rho \;\Rightarrow\; \min\left[\max(0,-\psi) - \nu\rho\right] \le \xi(\psi, X) \le \max\left[\max(0,-\psi) - \nu\rho\right]$$

$$\Rightarrow \quad 0 - \max(\nu.\rho) \le \xi(\psi, X) \le \left|\min(\psi)\right| - \min(\nu.\rho)$$

$$\Rightarrow -\nu.\sum_j \alpha_j . \max\left(\kappa(SV_c, SV_j)\right) \le \xi(\psi, X) \le \left|\min(\psi)\right| - \nu\sum_j \alpha_j . \min\left(\kappa(SV_c, SV_j)\right)$$

En utilisant le résultat (4.45) dans l'expression précédente, on peut écrire :

$$-\nu.\mathrm{Up}_{\ker} \le \xi(\psi, X) \le \mathrm{Up}_{\ker} - (1+\nu).\mathrm{Lo}_{\ker} \qquad 4.46$$

Avec le noyau gaussien : $\xi(\psi, X) \in [-\nu, 1], \quad 0 < \nu < 1$

b) Convergence du SAKM : Étude théorique de l'erreur de classification dynamique

On montre dans un premier temps qu'il existe une limite ℓ unique vers laquelle tend l'erreur de classification dynamique lorsque le temps devient suffisamment grand. Nous allons ensuite prouver la convergence de la procédure d'adaptation du SAKM en montrant que la limite ℓ est finie en déterminant ses bornes d'encadrement.

- Unicité de la limite ℓ : Il suffit de montrer que $\frac{\partial}{\partial t} E_{learn}[\psi^t] \underset{t\to\infty}{\to} 0$

$$\frac{\partial}{\partial t} E_{learn}[\psi^t] \approx \Delta E_{learn}[\psi^t] = \frac{1}{t}\sum_{i=1}^{t} \xi(\psi^i, X_i) - \frac{1}{t-1}\sum_{i=1}^{t-1} \xi(\psi^i, X_i)$$

$$= \left(\frac{1}{t(t-1)}\right).\left((t-1).\xi(\psi^t, X_t) - \left(\sum_{i=1}^{t-1} \xi(\psi^i, X_i)\right)\right)$$

En utilisant les bornes de la fonction coût ξ (4.46), on peut écrire :

$$\frac{1}{t}.\left(\min\left(\xi(\psi^t, X_t)\right) - \max\left(\xi(\psi^i, X_i)\right)\right) \le \Delta E_{learn}[\psi^t] \le \frac{1}{t}.\left(\max\left(\xi(f^t, X_t)\right) - \min\left(\xi(f^i, X_i)\right)\right)$$

$$\frac{1}{t}.\left((1+\nu).(\mathrm{Lo}_{\ker} - \mathrm{Up}_{\ker})\right) \le \Delta E_{learn}[\psi^t] \le \frac{1}{t}.\left((1+\nu).(\mathrm{Up}_{\ker} - \mathrm{Lo}_{\ker})\right)$$

Les bornes inférieure et supérieure ψ^t tendent vers 0 lorsque $t \to \infty$, d'où :

$$\lim_{t\to\infty}\left(\frac{\partial}{\partial t} E_{learn}[\psi^t]\right) = \lim_{t\to\infty}\left(E_{learn}[\psi^t] - E_{learn}[\psi^{t-1}]\right) \to 0 \qquad 4.47$$

On en déduit que E_{learn} devient stable (constante) lorsque $t \to \infty$

- Bornes de la limite ℓ : il s'agit de montrer que la limite ℓ est finie en la bornant

$$E_{learn}[\psi^t] = \frac{1}{t}\sum_{i=1}^{t} \xi\left(\psi^t, X_t\right) \Rightarrow \frac{1}{t}.t.\min\left(\xi(\psi^t, X_t)\right) \le E_{learn}[\psi^t] \le \frac{1}{t}.t.\max\left(\xi(\psi^t, X_t)\right)$$

$$-\nu.\mathrm{Up}_{\ker} \le E_{learn}[\psi^t] \le \mathrm{Up}_{\ker} - (1+\nu).\mathrm{Lo}_{\ker} \qquad 4.48$$

Avec le noyau gaussien : $E_{learn}[\psi^t] \in [-\nu, 1], \quad 0 < \nu < 1$

La moyenne des erreurs cumulées produites par l'algorithme SAKM converge vers une limite ℓ unique et bornée. Cette limite est non nulle mais se trouve dans un intervalle proche de 0. Ceci s'explique par le fait que la procédure d'adaptation du SAKM est une méthode approximative. Cette procédure itérative offre de bonnes propriétés de convergence comme cela est illustré sur la figure 4.13.

Remarque : La condition (4.48) n'est pas vérifiée par le NORMA et l'ALMA. En effet, les offsets ρ estimés par ces algorithmes ne sont pas bornés et peuvent diverger. Ceci est un inconvénient dans le contexte de l'apprentissage de cibles évolutives (figure 4.13).

- Généralisation des bornes de convergence dans un environnement multi-classe :

A partir de l'étude que nous venons de mener, on peut généraliser le résultat trouvé pour les problèmes multi-classes. Soit le modèle de classification $\Im^t = \{\psi_1^t, ..., \psi_M^t\}$ décrivant un nombre M (fini) de classes, la moyenne de l'erreur globale cumulée au cours l'adaptation des classes tend vers une limite finie et bornée. A partir de (4.48), on obtient le résultat suivant :

$$-\nu.M.\mathrm{Up}_{\ker} \le E_{learn}[\Im^t] \le M.\left(\mathrm{Up}_{\ker} - (1+\nu).\mathrm{Lo}_{\ker}\right) \qquad 4.49$$

Dans le cas du noyau gaussien : $E_{learn}[\psi^t] \in [-\nu.M, M], \quad 0 < \nu < 1$

c) Comparaison des algorithmes ALMA, NORMA et SAKM

Afin de comparer les performances des algorithmes ALMA, NORMA et SAKM dans le contexte du suivi de classes évolutives, des données non-stationnaires sont générées à partir d'une distribution gaussienne. On fait varier progressivement le centre de cette distribution de façon à obtenir une classe évolutive. La modélisation dynamique de cette classe est effectuée en utilisant chacun des trois algorithmes. Sur la figure 4.13, sont présentées les évolutions temporelles de l'offset ρ et l'erreur de classification dynamique E_{learn} produites par chacun de ces algorithmes.

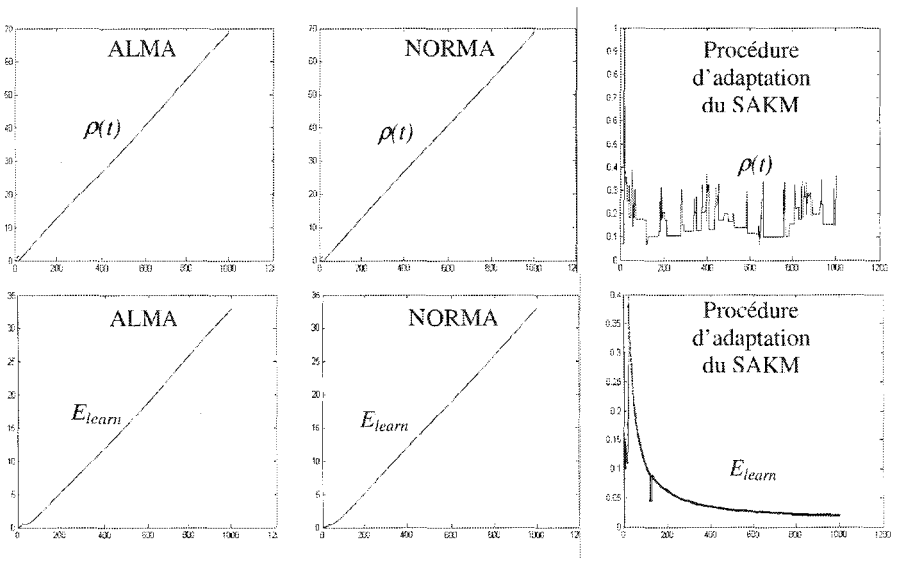

Figure 4.13 : Évolution temporelle de l'offset et la fonction objectif des algorithmes ALMA, NORMA et SAKM dans un problème de suivi d'une classe gaussienne évolutive.

Pour l'ALMA et le NORMA, l'offset ρ croît rapidement entraînant une augmentation importante de l'erreur E_{learn}. Ce phénomène est causé par les règles de mise à jour de ces algorithmes qui détectent un nombre trop important de données erreurs (*outliers*) et tentent de compenser leurs effets avec l'accroissement de l'offset. C'est un phénomène indésirable qui limite les performances de l'ALMA et du NORMA à la modélisation des classes statiques.

En revanche, l'offset ρ estimé par la procédure d'adaptation du SAKM reste faible. Cet offset est déterminé à partir de l'équation de l'hyperplan de la même façon que pour l'estimateur théorique One-Class-SVM. Par conséquent, on obtient une meilleure approximation du modèle au cours du suivi de classes évolutives et une bonne convergence de l'erreur de classification dynamique E_{learn} pour la procédure d'adaptation du SAKM. On constate, en effet, que la courbe de E_{learn} décroît progressivement et tend à converger vers une limite ℓ unique et bornée, comme cela a été démontré dans l'étude théorique menée dans la sous-section précédente.

4.3.5.2 Complexité du SAKM

A partir du calcul du nombre d'opérations de ses procédures d'apprentissage, nous déterminons la complexité du SAKM. Cette complexité dépend de la dimension D de données, du nombre maximum τ de vecteurs support dans chaque classe, du nombre de classes M et du nombre L de données à traiter. Du tableau 4.4, on déduit que la complexité du SAKM est de l'ordre de $O(D\times\tau^2\times M\times L)$. Le temps d'exécution de l'algorithme croît de façon polynomiale avec la dimension D, le carré du paramètre τ, le nombre de classes M et le nombre de données L. Dans la pratique la complexité est inférieure à celle déterminée par le calcul. En effet au cours de l'apprentissage, seul le critère de similarité est systématiquement appelé à chaque itération. L'utilisation des autres procédures dépend du résultat donné par le critère de similarité (voir tableau 4.3). Le degré de complexité de SAKM est satisfaisant dans la plupart des problèmes en ligne. Le SAKM est plus rapide que les algorithmes SVM incrémentaux [Cauwenberghs & al., 2001 ; Gretton & al., 2003].

Procédures d'apprentissage	Nb d'opérations	Complexité	Commentaires
Critère de Similarité	$(3D+5)\times\tau\cdot M+3D+6)\times L$	$O(D\times\tau\times M\times L)$	Complexité polynomiale
Procédure de Création	-	-	Affectations
Procédure d'Adaptation	$(3\tau D+6\tau+1)\times L$	$O(D\times\tau)$	Complexité polynomiale
Procédure de Fusion	$6\tau^2D+12\tau^2+2\tau$	$O(D\times\tau^2)$	Complexité polynomiale
Procédure d'Élimination	-	-	Suppressions

Tableau 4.4 : Calcul de la complexité des procédures d'apprentissage du SAKM

4.3.6 Discussion sur les paramètres du SAKM

L'algorithme SAKM est développé avec une procédure d'adaptation qui minimise un risque régularisé en utilisant la technique du gradient stochastique. Or, selon la théorie de Vapnik, un algorithme minimisant un tel risque limite les problèmes d'optima locaux. Mais, il est au préalable nécessaire d'effectuer un choix convenable des paramètres du SAKM. On distingue ces paramètres en deux groupes :

- *Les paramètre d'apprentissage (de modèle)* : le paramètre du noyau gaussien λ dont le choix se fait en fonction de la distribution des donnés, le ratio d'apprentissage

$\eta \in [0,1]$, le seuil de similarité $\varepsilon_{th} = 0,2$ déterminé par le calcul et τ la taille de la fenêtre de définition du modèle de classe.

- *Les paramètre de robustesse* : ce sont les mêmes que ceux de l'AUDyC N_{amb}, $N_{\min}$, T. En plus de ces paramètres, on a le cœfficient $\nu \in [0,1]$ qui est utilisé pour pondérer l'influence de l'offset dans l'erreur de classification dynamique.

Le paramètre λ dépend de la distribution des données. Son réglage est quelque fois difficile. Il peut se faire par essais et corrections à partir d'un échantillon de données. On peut également proposer une méthode d'initialisation de λ de façon similaire à celle proposée pour la matrice de covariance initiale de l'AUDyC. Dans le chapitre 5, partie 5.2, des expériences sont menées pour faciliter le réglage des paramètres de l'AUDyC et du SAKM.

4.4 Synthèse théorique et analyse comparée : AUDyC et SAKM

Après la présentation des algorithmes SAKM et AUDyC, une synthèse théorique est présentée dans cette section. Elle a pour but de comparer les deux algorithmes. Pour cela, on s'intéresse essentiellement aux caractéristiques les plus importantes que sont :

- **Le modèle de classification**

Le modèle de classification de l'AUDyC convient à la modélisation de données dont la densité peut être approximée par la loi gaussienne. C'est une modélisation paramétrique dont les performances sont étendues aux classes complexes en utilisant l'approche multimodale. Cependant, le modèle de classification de l'AUDyC n'est pas à l'abri des problèmes d'optima locaux. L'algorithme SAKM, quant à lui, utilise des fonctions d'apprentissage à noyau. Ces fonctions sont des modèles non-paramétriques capables de modéliser des classes complexes. Ces modèles ont donné des résultats très satisfaisants dans les problèmes de classification de données spatiales. D'un point de vue théorique, le modèle de classification du SAKM convient à la modélisation de classes de distribution quelconque. Quant à l'AUDyC, il convient particulièrement aux données gaussiennes.

- **La fonction objectif**

La fonction objectif utilisée pour l'AUDyC est un critère de type vraisemblance. Ce critère quantifie la qualité de la classification. La maximisation de ce critère par l'algorithme l'AUDyC garantit l'homogénéité des classes. Cependant, cette maximisation ne conduit à un bon résultat que si la densité des données suit la loi gaussienne. Dans le cas de l'algorithme SAKM, la fonction objectif est un critère de risque qui mesure instantanément l'erreur de classification dynamique. Le but est donc de minimiser ce critère de risque. Le risque utilisé est établi avec une fonction coût (risque empirique) et un terme de régularisation. Grâce à la régularisation, l'algorithme choisit le modèle adéquat parmi un ensemble de fonctions d'apprentissage afin d'éviter les problèmes optima locaux. Cette solution utilisée pour le SAKM est plus intéressante que celle de la vraisemblance. Cette dernière ne marche que si le modèle est bien choisi.

- **Paramètres de réglage**

Les algorithmes de classification dynamique que nous proposons sont critiquables pour leur nombre important de paramètres de réglage. L'algorithme AUDyC utilise 9 paramètres. Mais, seuls les paramètres de modèle posent quelques difficultés, en particulier le choix de la matrice de covariance initiale Σ_{ini}. Les autres paramètres sont fixés par expérience ou choisis en fonction du bruit dans les données et des objectifs du problème à traiter. Les paramètres du SAKM sont également nombreux (7 paramètres au total). Le SAKM utilise moins de paramètres que l'AUDyC. Le choix du paramètre du noyau gaussien λ peut être délicat lorsque la distribution des données n'est pas connue.

- **Processus d'apprentissage**

Afin de conférer aux algorithmes de classification dynamique des capacités d'auto-adaptation en environnement non-stationnaire, nous avons proposé quatre procédures d'apprentissage. Ces procédures sont développées pour l'AUDyC avec des règles d'apprentissage établies à partir du modèle gaussien. Les éléments mathématiques de ce modèle étant bien maîtrisés et très utilisés, des travaux antérieurs ont montré les bonnes performances de la plupart des règles (adaptation, fusion) utilisées par l'AUDyC [Lurette, 2003]. Quant au SAKM, le développement de ses règles d'apprentissage convenables n'est pas une tâche facile. Cet algorithme est, en effet, basé sur des techniques plus récentes pour lesquelles très peu de travaux sont effectués dans le contexte de l'apprentissage en ligne. A cause de ces difficultés, nous n'avons pas réussi à mettre en œuvre un mécanisme de scission pour le SAKM pouvant fonctionner dans les problèmes en ligne. Le processus d'apprentissage de l'AUDyC est (relativement) plus complet que celui du SAKM. En effet, contrairement au SAKM, l'AUDyC dispose d'un mécanisme de scission de classes visant à séparer les groupes de prototypes distants dans une même classe.

- **Convergence**

La convergence est une propriété très importante pour les classifieurs dynamiques. Afin de prouver la convergence de nos algorithmes, nous nous sommes particulièrement intéressés à la procédure d'adaptation récursive. En effet, cette procédure est le moteur de l'algorithme et est déterminante sur les résultats successifs produits par l'algorithme. On suppose que les mécanismes de fusion et de scission de classes ne causent pas de défauts majeurs de modélisation. La procédure d'adaptation récursive de l'AUDyC est basée sur une règle de mise à jour exacte. C'est donc une procédure convergente. La convergence du SAKM, quant à elle, est démontrée à travers l'analyse théorique de l'erreur de classification dynamique. Cette erreur tend vers une limite ℓ unique et bornée dans un intervalle autour de 0. On en déduit la convergence du SAKM. La procédure d'adaptation du SAKM produit des modèles approximatifs alors que celle de l'AUDyC estime les modèles exacts et converge vers 0. Précisons que ces propriétés de convergence sont des conditions nécessaires mais non suffisantes pour garantir la convergence de nos classifieurs dynamiques.

- **Complexité algorithmique**

Dans les applications en ligne, la complexité de l'algorithme est l'un des facteurs les plus importants pour un algorithme de classification dynamique. Cette complexité, si elle est trop importante, restreint les possibilités d'utilisation de l'algorithme à quelques problèmes temps-réel. En effet, cette grandeur détermine la rapidité de l'algorithme et les ressources à allouer. La complexité de l'AUDyC est de $O(D^3 \times J^2 \times L)$. Cela signifie que le temps de calcul de l'algorithme augmente de façon polynomiale en fonction de la dimension (au cube), du nombre de prototypes (au carré) et du nombre de données. Concernant le SAKM, sa complexité est de $O(D \times \tau^2 \times M \times L)$. C'est également une complexité polynomiale qui croît avec la dimension, le nombre de vecteurs support (au carré) et le nombre de données. Il est intéressant de constater que le SAKM est plus adapté aux données de grande dimension que l'AUDyC.

Cette analyse comparée est donnée uniquement d'un point de vue théorique. Dans le chapitre suivant, les algorithmes sont expérimentés sur des données de simulation et applications réelles. Ces expériences permettront d'apprécier les performances des algorithmes.

4.5 Conclusion

Le développement d'algorithmes de classification dynamique présente des difficultés variées. Certaines de ces difficultés sont fortement liées à la nature du problème à traiter, il n'est donc pas toujours facile de proposer des solutions fonctionnelles et généralisables dans toutes les applications. La description générique proposée dans le chapitre 1 présente l'intérêt de faciliter la conception de classifieurs dynamiques. Cette tâche requiert le choix d'un modèle de classification adéquat pour représenter les classes et la mise en œuvre de règles auto-adaptatives utiles pour l'apprentissage en environnement dynamique. Le développement de ces règles peut être plus ou moins compliqué selon le modèle de classification choisi.

En utilisant une modélisation avec un mélange gaussien suivant une approche multimodale, l'algorithme AUDyC donne une approximation correcte des classes complexes. Cependant, c'est une méthode de modélisation qui ne convient qu'à des données supposées gaussiennes. De même, elle ne garantit pas à l'algorithme d'être à l'abri des problèmes d'optima locaux. Quant à l'algorithme SAKM, il utilise des fonctions d'apprentissage à base de noyau pour représenter les classes. Ce sont des modèles non-paramétriques capables de représenter des classes de distribution quelconque. Ces deux algorithmes sont dotés de règles d'auto-adaptation leur permettant de prendre en compte les modifications et les évolutions de classes en environnement dynamique. A partir d'une étude théorique, nous avons montré les bonnes propriétés de convergence de ces algorithmes. De même, leurs complexités

algorithmiques ont été calculées. Ces complexités polynomiales sont satisfaisantes dans la plupart des applications en ligne. On note que lorsqu'il s'agit de données de grande dimension, la complexité de l'algorithme AUDyC croît plus rapidement que celle de l'algorithme SAKM.

Le chapitre suivant est consacré aux résultats d'expérimentation des algorithmes AUDyC et SAKM. Ces expériences sont, en grande partie, réalisées sur de données de simulation. On y présente également quelques réalisations de systèmes d'aide à la décision. Parmi ces réalisations, une application réelle de surveillance d'un processus est exposée.

Chap. 5 : Expérimentation de l'AUDyC et du SAKM

5.1 Introduction

Dans ce chapitre, nous exposons les résultats expérimentaux issus de plusieurs tests effectués avec les deux algorithmes présentés au chapitre 4. Les objectifs visés sont multiples : réglage des paramètres, expérimentation des procédures d'apprentissage, analyse des performances (qualité de modélisation, complexité), validation sur des applications réelles.

Avant d'exposer et d'analyser les résultats expérimentaux, il est au préalable nécessaire de rappeler quelques aspects théoriques sur lesquels sont fondés nos algorithmes de classification dynamique. Le modèle de classification de l'algorithme AUDyC est un modèle de mélange inspiré du mélange gaussien. Tandis que le SAKM utilise les fonctions d'apprentissage à noyau pour modéliser les classes. Pour chacun de ces deux algorithmes, la qualité de modélisation de la partition dynamique dépend du réglage de certains paramètres dits de modèle. Plusieurs paramètres sont introduits lors de la conception de ces classifieurs dynamiques, mais peu de ces paramètres sont difficiles à régler. A travers les expériences menées dans ce chapitre, on montre que la plupart de ces paramètres ont pour rôle essentiel d'accroître la flexibilité et la robustesse des algorithmes AUDyC et SAKM.

Par ailleurs, afin de conférer à ces classifieurs des capacités d'auto-adaptation, leurs processus d'apprentissage sont élaborés en quatre procédures principales : **création, adaptation, fusion** et **évaluation**. La procédure d'évaluation de l'AUDyC est élaborée avec un mécanisme de scission et d'élimination tandis celle du SAKM ne comporte qu'un mécanisme d'élimination. Chacune de ces procédures a pour rôle de traiter un phénomène dynamique (apparition de classes, évolution de classes, fusion de classes,...) dû à la non-stationnarité des données. Dans ce chapitre, l'efficacité de ces procédures d'apprentissage des algorithmes AUDyC et SAKM est évaluée sur des données artificielles créées à cet effet. A

travers ces simulations, on illustre les capacités de modélisation adaptative de données non-stationnaires et de suivi de classes évolutives.

Les performances théoriques des algorithmes AUDyC et SAKM ont été étudiées dans le chapitre 4 (sections 4.2.5 et 4.3.5). Cette étude a montré que la procédure d'adaptation récursive de l'AUDyC converge vers 0 (méthode de mise jour exacte), tandis que celle du SAKM converge vers une limite ℓ unique et bornée (méthode de mise à jour approximative). Des tests de convergence effectués sur des données de simulation ont confirmé ces résultats théoriques. Enfin, en calculant le nombre d'opération de chaque algorithme, la complexité algorithmique a été évaluée; elle est de l'ordre de $O(D^3 \times J^2 \times L)$ pour l'AUDyC (chapitre 4, tableau 4.2) et de $O(D \times \tau^2 \times M \times L)$ pour le SAKM (chapitre 4, tableau 4.4).

Les résultats expérimentaux sont présentés sous forme de figures accompagnées de tableaux d'indicateurs et/ou de courbes. Les figures nous donnent une appréciation visuelle de la qualité de modélisation des algorithmes. Parmi les indicateurs présentés dans les tableaux ou sous forme graphique, on trouve la fonction objectif (vraisemblance ou risque), le temps d'exécution, le pourcentage de classification, etc. Ces indicateurs sont choisis en fonction des intérêts du test et sont des mesures (quantitatives) de performances. Les conditions de réalisation de chaque expérience sont d'abord exposées avant d'analyser et interpréter les résultats produits par chacun des algorithmes.

Le présent chapitre est organisé en cinq parties :

- La première partie (partie 5.2) est consacrée à l'analyse de l'influence des paramètres des algorithmes dans l'objectif de faciliter leur réglage.
- La seconde partie (partie 5.3) expérimente les procédures d'apprentissage des deux algorithmes. Des données artificielles sont créées à cet effet, pour décrire les différents scénarios de la problématique de classification dynamique (chapitre 1, partie 1.2).
- Les tests de la troisième partie (partie 5.4) sont menés pour apprécier la complexité algorithmique de l'AUDyC et du SAKM d'un point de vue pratique.
- A partir de ces résultats de simulation, la partie 5.5 expose une synthèse pratique visant à comparer les performances des deux algorithmes.
- Dans la dernière partie (partie 5.6), nous proposons de concevoir deux Systèmes d'Aide à la Décision (SAD) basés sur nos algorithmes de classification dynamique. Le premier SAD est dédié à la surveillance en ligne d'un thermorégulateur tandis que le second est proposé pour le suivi en ligne de processus non-stationnaires.

5.2 Influence et réglage des paramètres des algorithmes

Dans cette partie, plusieurs tests sont effectués sur des données de simulation afin d'analyser l'influence des paramètres des algorithmes sur la qualité du modèle de classification. Les tests sont réalisés en utilisant les mêmes données et dans les mêmes conditions pour l'AUDyC et pour le SAKM. Les données utilisées pour chaque test sont soigneusement générées de façon à décrire des classes statiques ou des classes évolutives selon la nature et l'influence du paramètre à analyser. Chaque test vise à analyser l'influence d'un paramètre à la fois sur le modèle de classification. Pour cela, on fait varier la valeur ce seul paramètre pendant que tous les autres paramètres sont maintenus fixes. Les tests effectués dans cette partie ont pour intérêt de faciliter le réglage des paramètres de l'AUDyC et ceux du SAKM afin de permettre l'utilisation de ces algorithmes dans divers problèmes de classification dynamique.

Les paramètres de robustesse sont communs aux deux algorithmes (chapitre 4, sections 4.2.6 et 4.3.6). Ce sont notamment le seuil d'ambiguïté N_{amb}, le seuil de cardinalité $N_{\min}$ et le nombre d'échantillons T entre deux procédures d'évaluation. Le réglage de ces paramètres dépend en réalité du bruit dans les données. Leur influence sera étudiée dans l'expérimentation des procédures de fusion et d'élimination (partie 5.3). Dans cette partie, on s'intéresse particulièrement aux paramètres dits de modèle dont le réglage est quelquefois difficile. En effet, ces paramètres dépendent pour la plupart de la distribution des données.

La présente partie est divisée en deux sections. L'influence et le réglage des paramètres de l'AUDyC sont étudiés dans la section 5.2.1. La même étude est effectuée pour les paramètres du SAKM dans la section 5.2.2. En réalité, plusieurs tests sont effectués pour maîtriser le réglage de ces paramètres, seuls quelques uns sont présentés dans cette partie.

5.2.1 Influence et réglage des paramètres de l'AUDyC

L'utilisation de l'AUDyC dans les meilleures conditions nécessite un bon réglage de ses paramètres. Les tests effectués dans cette section visent d'une part, à montrer l'influence des paramètres et d'autre part, à tirer les conclusions utiles pour faciliter le réglage de ces paramètres à partir de l'analyse des résultats obtenus. Rappelons que les paramètres de modèle de l'AUDyC sont les suivants :

- le paramètre σ_{ini} initialisant la matrice de covariance des prototypes,
- les seuils $\mu_{\min}$ et $\mu_{\max}$ d'appartenance respectifs aux prototypes et aux classes,
- la taille N_P de la fenêtre de définition du prototype gaussien.

5.2.1.1 Sensibilité de la matrice de covariance initiale σ_{ini}

Le paramètre σ_{ini} est utilisé pour initialiser la matrice de covariance de prototypes : $\Sigma_{ini} = \sigma_{ini}^2 I_D$ (I_D : matrice identité de dimension D). Ce paramètre fixe la taille de l'hyper-volume des nouveaux prototypes. La maîtrise du réglage de ce paramètre est d'une importance considérable. En effet, σ_{ini} influe sur la convergence et la rapidité de l'algorithme.

Conditions de réalisation : L'expérience est réalisée avec des classes statiques afin d'analyser localement et appréhender l'influence du paramètre σ_{ini}. 1000 données de simulation sont créées pour former 3 classes : 2 classes gaussiennes (mono-prototypes) et 1 classe complexe composée de 2 prototypes. Plusieurs tests sont effectués avec des tirages aléatoires. Les données sont présentées séquentiellement à l'AUDyC. L'expérience est réalisée pour 4 valeurs de σ_{ini} ; tous les autres paramètres étant maintenus fixes.

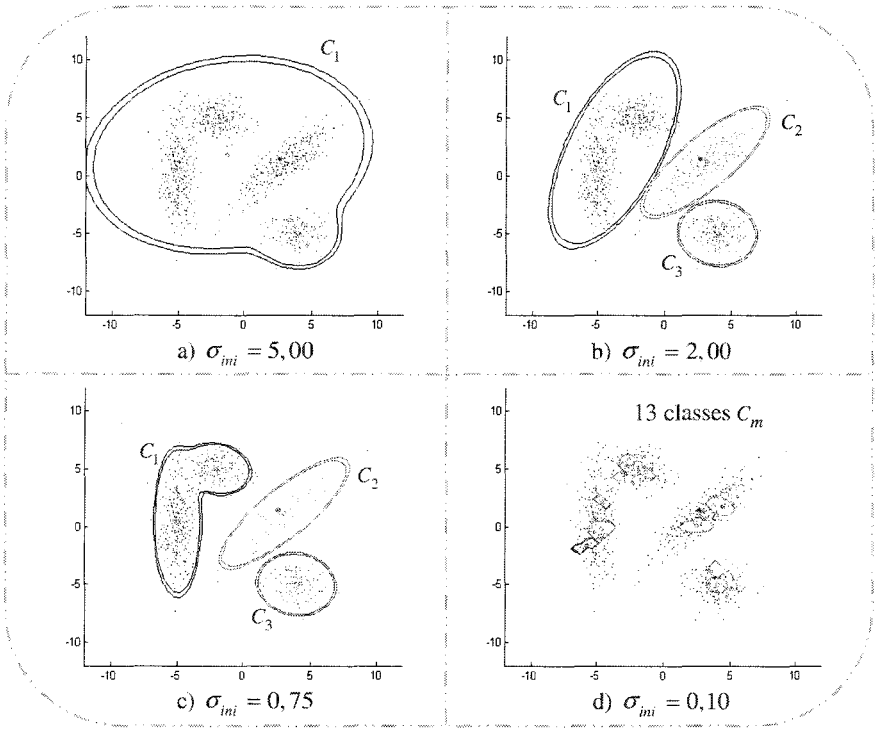

Figure 5.2.1 : Influence du paramètre σ_{ini} sur le modèle de classification de l'AUDyC. Paramètres fixés : $\mu_{\min} = 0{,}02$, $\mu_{\max} = 0{,}03$, $N_P = \infty$, $N_{\min} = 10$, $N_{amb} = 10$, $T = 333$.

Paramètre σ_{ini}	**5,00**	**2,00**	**0,75**	**0,10**
Vraisemblance L_{poss}	-2,66	-1,27	-0,72	0,05
Pourcent X_i *classées*	100,00	97,30	99,70	23,60
Tps d'exécution (sec.)	8,43	8,03	10,64	532,50
Nb classes (prototypes)	1(2)	3(3)	3(4)	13(17)

Tableau 5.2.1 : Indicateurs de performance de l'AUDyC pour 4 valeurs du paramètre σ_{ini}.

Interprétation des résultats : Les résultats de l'expérience sont présentés sur la figure 5.2.1. Quelques indicateurs de performance sont également proposés dans le tableau 5.2.1. Ces indicateurs sont des valeurs moyennes estimées à partir de plusieurs essais avec des tirages différents. Lorsque σ_{ini} est trop grand, les prototypes sont initialisés avec un volume trop important. Dans ces conditions, une seule classe est créée pour contenir toutes les données (figure 5.2.1.a). A l'inverse, si σ_{ini} est choisi trop petit, plusieurs petits prototypes et classes sont créés (figure 5.2.1.d). Le paramètre $\sigma_{ini} = 0,75$ conduit à un modèle de classification satisfaisante pour les présentes données (figure 5.2.1.c). Dans le tableau 5.2.1, ce réglage offre un pourcentage de classification important avec une vraisemblance possibiliste relativement élevée. Dans la dernière colonne du tableau 5.2.1, on a la vraisemblance maximale, mais pour un pourcentage de classification médiocre. Un autre phénomène intéressant révélé par ce test est l'augmentation considérable du temps de calcul pour des valeurs trop petites de σ_{ini} (tableau 5.2.1). En effet, les petites valeurs de σ_{ini} entraînent la création d'un grand nombre de prototypes et accroissent ainsi la complexité algorithmique de l'AUDyC.

Le choix du σ_{ini} peut se faire par essais et corrections à partir d'un échantillon de données préalablement extraites. Afin de faciliter le choix du paramètre σ_{ini}, nous proposons une méthode d'initialisation en Annexe A.4. Cette méthode est basée sur l'analyse de la distribution des données pour estimer le paramètre σ_{ini}. Par exemple, pour les données de la présente expérience, on trouve $\sigma_{ini} = 0,42$. Cette valeur n'est certes pas la meilleure trouvée par la simulation. Néanmoins, cette méthode d'initialisation permet de trouver une valeur approchée permettant un réglage plus rapide.

5.2.1.2 Sensibilité des paramètres $\mu_{\min}$ et $\mu_{\max}$

Les paramètres $\mu_{\min}$ et $\mu_{\max}$ représentent les seuils d'appartenance respectifs aux prototypes et aux classes. Ces seuils sont utilisés par le critère de similarité lors de la classification des données. Afin d'étudier l'influence des deux seuils, deux expériences sont réalisées : La première vise à analyser l'influence de l'écart $\mu_{\max} - \mu_{\min}$, tandis que dans le seconde, l'intérêt est porté sur l'influence de l'accroissement simultané des deux seuils $\mu_{\min}$ et $\mu_{\max}$ sur la qualité du modèle de classification.

Conditions de réalisation : Nous utilisons les mêmes données du test de la section 5.2.1.1. Dans la première expérience, la variation progressive de l'écart $\mu_{\max} - \mu_{\min}$ est effectuée en maintenant fixe le seuil $\mu_{\max} = 0,02$ et en diminuant progressivement le seuil $\mu_{\min}$. Dans le seconde expérience, les seuils $\mu_{\min}$ et $\mu_{\max}$ sont augmentés simultanément. Au cours de ces tests, tous les autres paramètres de l'AUDyC restent inchangés.

Interprétation des résultats : Les résultats de la première expérience sont présentés sur la figure 5.2.2 et dans le tableau 5.2.2. Si l'écart $\mu_{\max} - \mu_{\min} = 0$, il n'y a pas de fusion de

classes car aucune donnée ambiguë ne peut se trouver entre les seuils μ_{max} et μ_{min} (chapitre 4, section 4.2.3.3). La figure 5.2.2 met en évidence le chevauchement de deux classes mono-prototypes au lieu de leur fusion en une seule classe. Par contre, si l'écart $\mu_{max} - \mu_{min}$ est trop grand, la fusion de classes est favorisée au détriment de celle des prototypes. En effet, les prototypes créés sont trop petits (figure 5.2.2.d) et ont donc peu de chance de partager des données ambiguës. Par ailleurs, ce type de réglage favorise la création de prototypes dans chaque classe. Le tableau 5.2.2 montre une augmentation du temps de calcul de l'algorithme causée par la création d'un nombre important de prototypes. Pour $\mu_{min} = 0,02$ et $\mu_{max} = 0,03$ ($\mu_{max} - \mu_{min} = 0,01$), on obtient une meilleure qualité de modélisation (figure 5.2.2.c).

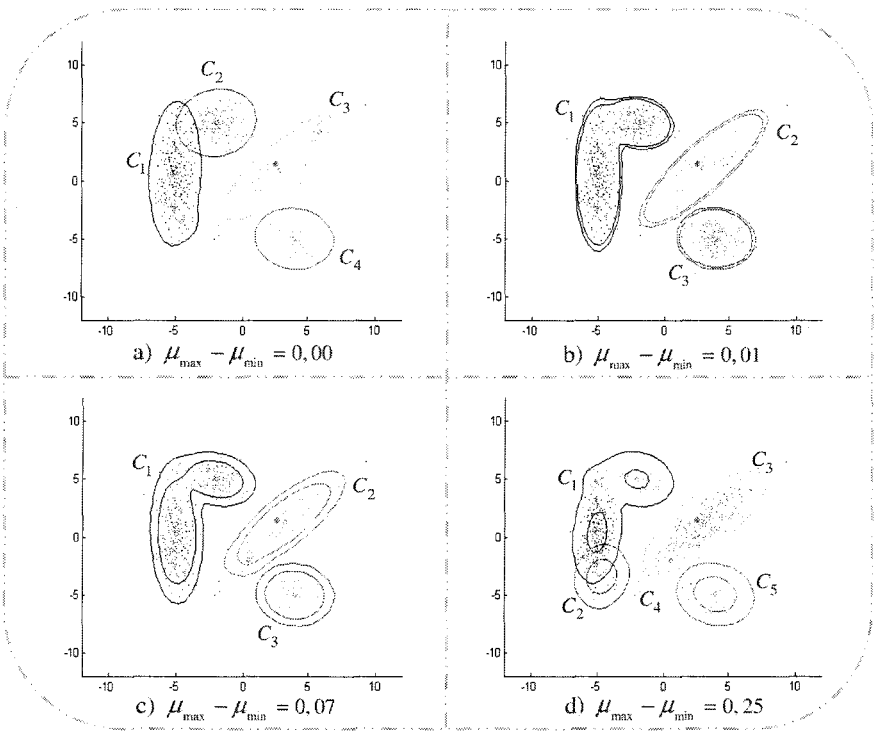

Figure 5.2.2 : Influence de μ_{min} et μ_{max} sur le modèle de classification de l'AUDyC : variation de $\mu_{max} - \mu_{min}$. Paramètres fixés : $\sigma_{ini} = 0,8$; $\mu_{min} = 0,02$; $N_P = \infty$; $N_{min} = 10$; $N_{amb} = 10$; $T = 333$.

Ecart $\mu_{max} - \mu_{min}$	**0,00**	**0,01**	**0,07**	**0,25**
Vraisemblance L_{poss}	-0,38	-1,03	-0,59	-0,32
Pourcent X_i *classées*	98,80	98,90	95,30	92,21
Tps d'exécution (sec.)	52,36	76,65	101,26	314,18
Nb classes (prototypes)	4(4)	3(4)	3(4)	5(7)

Tableau 5.2.2 : Indicateurs de performance de l'AUDyC en fonction de l'écart $\mu_{max} - \mu_{min}$.

La figure 5.2.3 et le tableau 5.2.3 présentent les résultats du second test pendant lequel les seuils μ_{min} et μ_{max} varient simultanément. On constate que ces résultats ont de fortes

similitudes avec ceux obtenus lors du test effectué sur l'influence de σ_{ini} (section 5.2.1). Pour des valeurs trop petites des seuils $\mu_{\min}$ et $\mu_{\max}$, toutes les données sont englobées dans une seule classe (figure 5.2.3). A l'inverse, si ces seuils sont choisis trop grands, plusieurs prototypes et classes sont localement créés. La meilleure partition est trouvée en fixant $\mu_{\min} = 0,02$ et $\mu_{\max} = 0,03$ (figure 5.2.2).

On montre dans l'Annexe A.4 que les seuils $\mu_{\min}$ et $\mu_{\max}$ ne dépendent pas de la distribution des données. Dans notre étude, on peut fixer définitivement ces seuils et porter les efforts de réglage sur le paramètre σ_{ini}. Pour toutes les simulations réalisées dans les prochaines expériences, les seuils d'appartenance sont fixés à $\mu_{\min} = 0,02$ et $\mu_{\max} = 0,03$.

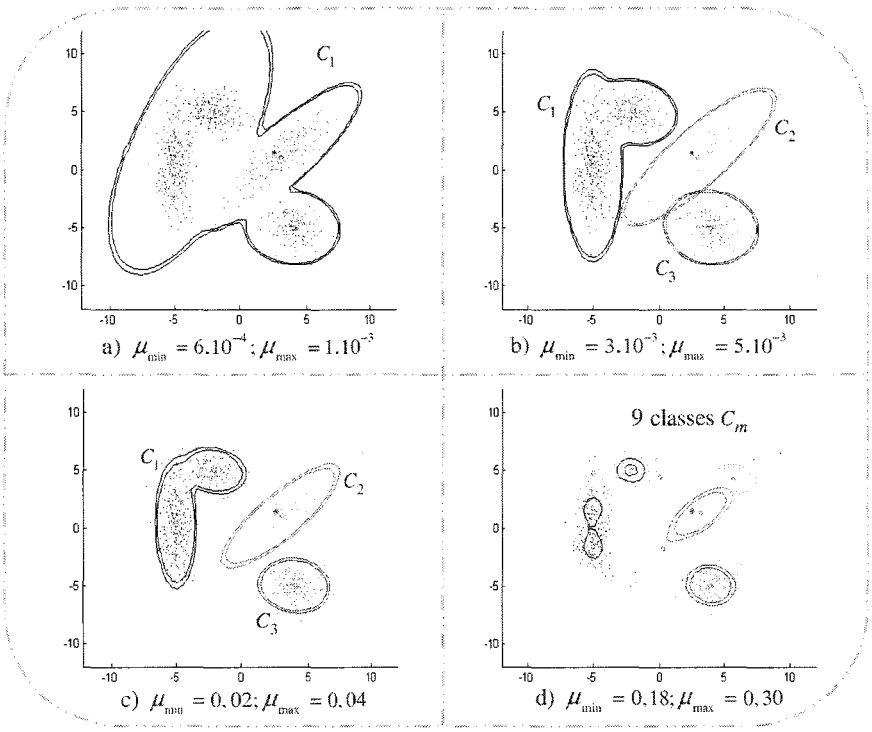

Figure 5.2.3 : Influence de $\mu_{\min}$ et $\mu_{\max}$ sur le modèle de classification de l'AUDyC : variation simultanée de $\mu_{\min}$ et $\mu_{\max}$. Paramètres fixés : $\sigma_{ini} = 0,8$, $N_P = \infty$, $N_{\min} = 10$, $N_{amb} = 10$, $T = 333$.

Seuil $\mu_{\max}$	**1.10^{-3}**	**5.10^{-3}**	**0,04**	**0,30**
Seuil $\mu_{\min}$	**6.10^{-4}**	**3.10^{-3}**	**0,02**	**0,18**
Vraisemblance L_{poss}	-0,31	-1,75	-0,88	-0,18
Pourcent X_i *classées*	100,00	99,80	99,90	96,00
Tps d'exécution (sec.)	8,67	10,33	53,14	86,25
Nb classes (prototypes)	1(3)	3(4)	3(4)	9(9)

Tableau 5.2.3 : Indicateurs de performance de l'AUDyC en fonction de la variation simultanée de seuils $\mu_{\min}$ et $\mu_{\max}$.

5.2.1.3 Sensibilité du paramètre N_P

Le paramètre N_P correspond à la taille de la fenêtre de définition des prototypes. Il joue un rôle important dans les problèmes de suivi de classes évolutives. En effet, le paramètre N_P est utilisé par l'AUDyC pour effectuer la mise à jour récursive de classes à travers l'ajout et le retrait d'informations. L'influence de ce paramètre est analysée dans cette section afin de faciliter son réglage.

Conditions de réalisation : Cette expérience est réalisée en utilisant des données non-stationnaires décrivant 2 classes évolutives. Chacune de ces 2 classes est constituée de 1000 données générées avec la loi gaussienne. L'évolution des classes est obtenue en déplaçant progressivement la moyenne et en augmentant le volume de chaque classe. L'algorithme AUDyC est utilisé pour effectuer plusieurs tests en variant le paramètre N_P.

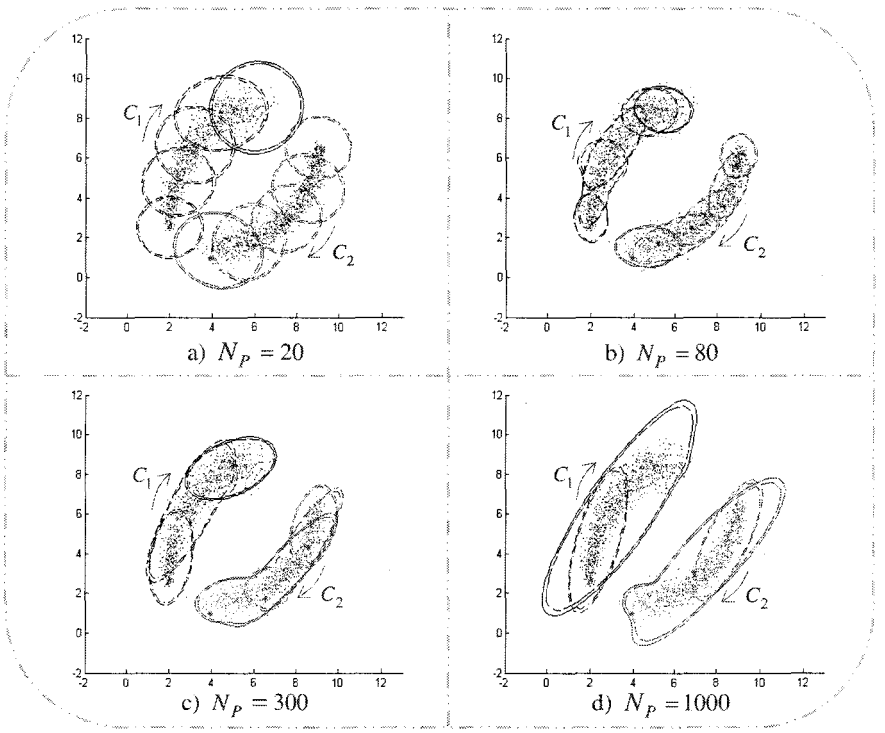

Figure 5.2.4 : Influence du paramètre N_P sur la classification dynamique de l'AUDyC. Paramètres fixés : $\sigma_{ini} = 0{,}8$, $\mu_{\min} = 0{,}02$, $\mu_{\max} = 0{,}03$, $N_{\min} = 10$, $N_{amb} = 10$, $T = 500$.

Paramètre N_P	**20**	**80**	**300**	**1000**
Tps d'exécution (sec.)	21.23	21.84	20.05	19.26
Nb classes (prototypes)	2(2)	2(2)	2(2)	2(3)

Tableau 5.2.4 : Indicateurs de performance de l'AUDyC en fonction du paramètre N_P.

Interprétation des résultats : Lorsque le paramètre N_P est fixé trop petit, le nombre de données dans les prototypes est insuffisant pour faire converger leurs modèles (figure 5.2.4.a).

Plus précisément, la cardinalité de chaque prototype est trop petite pour que la règle d'adaptation puisse estimer le modèle optimal à partir de l'initialisation (volume initial trop grand). On peut résoudre ce problème en diminuant la valeur de σ_{ini}. A l'opposé, si le paramètre N_P est choisi trop élevé, il y a un risque considérable de sur-apprentissage si la distribution de ces données n'est pas gaussienne (figure 5.2.4.d). Pour ce test, seuls le temps de calcul et le nombre de prototype et de classes sont présentés dans le tableau 5.2.4, les autres indicateurs n'apportent pas d'information utile pour le réglage de N_P. On remarque que le paramètre N_P n'influe pas sur le temps d'exécution de l'AUDyC. Le réglage de ce paramètre décide de l'ancienneté des données à retirer de l'apprentissage. Or, l'ancienneté est une notion subjective qui nécessite des connaissances *a priori* sur le problème à traiter.

Nous venons d'analyser la sensibilité et les conditions de réglage des paramètres de modèle sur la classification dynamique de l'AUDyC. L'influence des autres paramètres est montrée au cours de l'expérimentation des procédures d'apprentissage dans la partie 5.3.

5.2.2 Influence et réglage des paramètres du SAKM

Comme cela a été précédemment effectué pour l'AUDyC, nous proposons les mêmes tests pour les paramètres du SAKM. Ces tests visent principalement à faciliter le réglage des paramètres de modèle à partir de l'analyse de leur influence. Les tests sont effectués pour les paramètres suivants :

- le paramètre λ du noyau gaussien,
- le ratio d'apprentissage $\eta \in [0.1]$,
- la taille τ de la fenêtre exponentielle de définition du modèle de classe.

5.2.2.1 Sensibilité du paramètre du noyau gaussien λ

Le paramètre λ du noyau gaussien dépend de la distribution des données. Le réglage de ce paramètre a une influence prépondérante sur la qualité de modélisation.

Conditions de réalisation : Cette expérience est réalisée en utilisant les mêmes données de la section 5.2.1.1. Rappelons que ces données décrivent 1 classe complexe et 2 classes gaussiennes. La partition de ces données est estimée par l'algorithme pour plusieurs valeurs de λ en maintenant tous les autres paramètres fixes.

Interprétation des résultats[1] : La figure 5.2.5 présente les résultats de l'expérience. Pour des valeurs très petites de λ, une seule classe est créée (figure 5.2.5.a). En effet, toutes les données deviennent similaires car $\lambda \to 0$ implique $\mu\phi \to 1$ (chapitre 4, section 4.3.2). Et à l'opposé, pour des très grandes valeurs de λ, les données sont orthogonales (dissimilaires).

[1] Un modèle de classe du SAKM est une fonction d'apprentissage à noyau qui peut être représentée dans l'espace de données par plusieurs contours disjoints. Le modèle gaussien n'ayant pas cette propriété, chaque classe de l'AUDyC est représentée par un unique contour fermé dans l'espace.

Dans le tableau 5.2.5, avec $\lambda = 0.8$ et $\lambda = 1$, on obtient 3 classes avec un risque d'apprentissage faible et un pourcentage de classification élevé. La modélisation est meilleure pour ces réglages (figures 5.2.5.b et 5.2.5.c). Par ailleurs, on constate que le temps d'exécution croît avec λ. Ceci est du au fait que les grandes valeurs de λ favorisent la création de classes et de vecteurs support et par conséquent accroissent la complexité du SAKM.

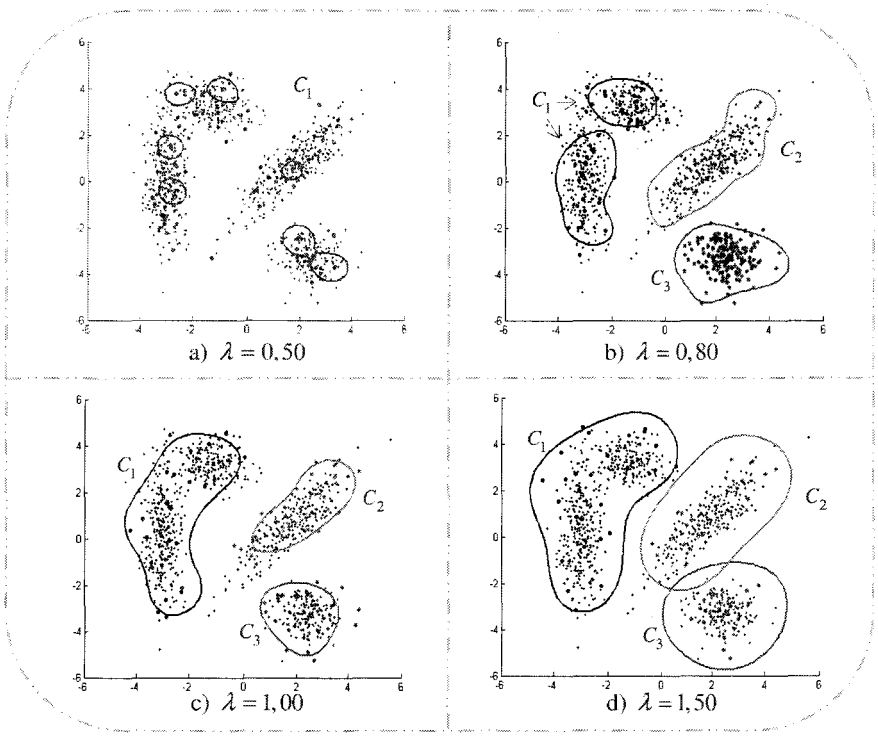

Figure 5.2.5 : Influence de λ sur le modèle de classification du SAKM. Paramètres fixés : $\nu = 0,3$, $\eta = 0,1$, $\tau = 30$, $N_{\min} = 10$, $N_{amb} = 10$, $T = 333$.

Dans la plupart des problèmes, le réglage de λ peut se faire par essais et corrections à partir d'un échantillon de données préalablement extraites.

Paramètre λ	**0,50**	**0,80**	**1,00**	**1,50**
Risque E_{learn}	0,12	0,20	0,22	0,19
Pourcent X_i *classées*	70%	92%	94%	91%
Tps d'exécution (sec.)	3,99	5,80	5,83	6,17
Nb classes (SVs)	1(30)	3(90)	3(90)	3(90)

Tableau 5.2.5 : Indicateurs de performance du SAKM en fonction du paramètre λ.

5.2.2.2 Sensibilité du ratio d'apprentissage η

Le ratio d'apprentissage η joue un rôle important dans la procédure d'adaptation du SAKM. Dans le cadre de la classification dynamique, le ratio est choisi dans l'intervalle $]0,1[$.

Conditions de réalisation : Cette expérience est réalisée avec des données non-stationnaires décrivant deux classes évolutives. Nous utilisons les mêmes données que celles utilisées dans la section 5.2.1.3.

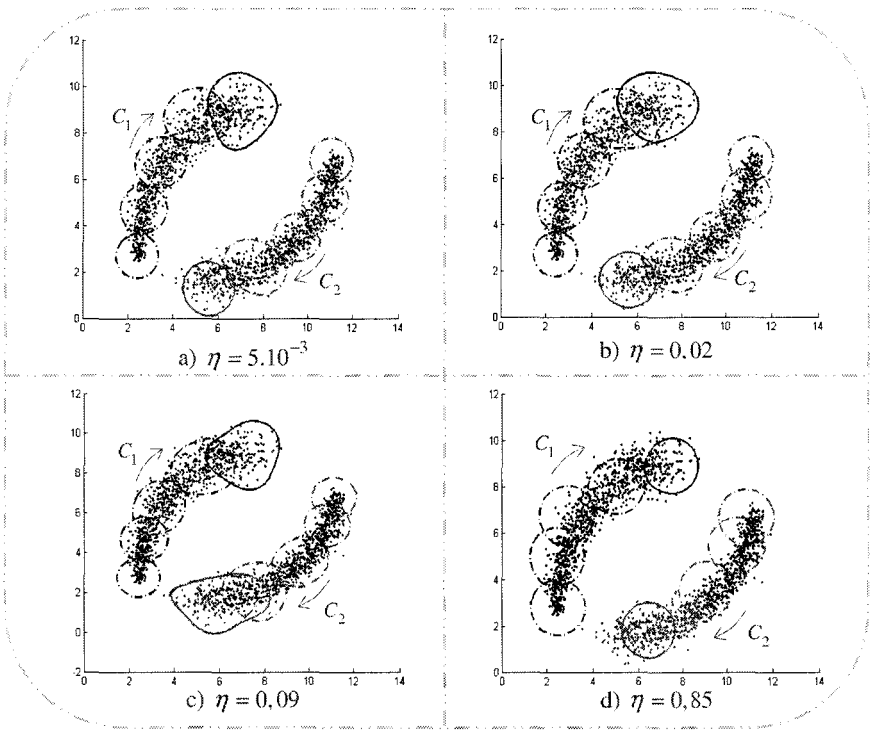

Figure 5.2.6 : Influence de η sur le modèle de classification du SAKM. Paramètres fixés : $\lambda = 0,8$, $\nu = 0,3$, $\tau = 30$, $N_{\min} = 10$, $N_{amb} = 10$, $T = 333$.

Paramètre η	**5.10^{-3}**	**0,02**	**0,09**	**0,85**
Tps d'exécution (sec.)	5,53	5,56	5,44	5,04
Nb classes (SVs)	2/60	2/60	2/60	2/20

Tableau 5.2.6 : Indicateurs de performance du SAKM en fonction du paramètre η.

Interprétation des résultats : Si le ratio η est choisi trop petit, l'adaptation des modèles est moins sensible aux variations brusques (figure 5.2.6.a). Par contre, si ce ratio est proche de 1, le modèle devient très réactif, il y a une forte sensibilité au bruit (figure 5.2.7.d). Le modèle de chaque classe est défini sur un plus petit nombre de vecteurs support, i.e. les plus récents (tableau 5.2.6). En effet, l'influence des vecteurs support est très vite atténuée. Le tableau 5.2.6 montre également que η n'agit pas sur le temps d'exécution du SAKM.

Le choix du paramètre η ne dépend pas de la distribution des données, mais plutôt de l'importance que l'on souhaite accorder aux informations les plus récentes par rapport aux plus anciennes. Par ailleurs, un réglage adéquat de ce paramètre permet d'atténuer l'influence du bruit. Dans la plupart des simulations, on fixe le ratio d'apprentissage $\eta \approx 0,1$.

5.2.2.3 Sensibilité du paramètre τ

Le paramètre τ fixe le nombre maximal de vecteurs support utilisés pour modéliser chaque classe. τ représente la taille de la fenêtre exponentielle décrite par les poids α_i des vecteurs support SV_i dans le modèle de classe. Afin de maîtriser le réglage de ce paramètre, nous analysons son influence dans le cadre du suivi de classes évolutives.

Conditions de réalisation : L'expérience est réalisée avec les mêmes données que celles de l'expérience précédente. Plusieurs tests sont effectués en ne variant que le paramètre τ.

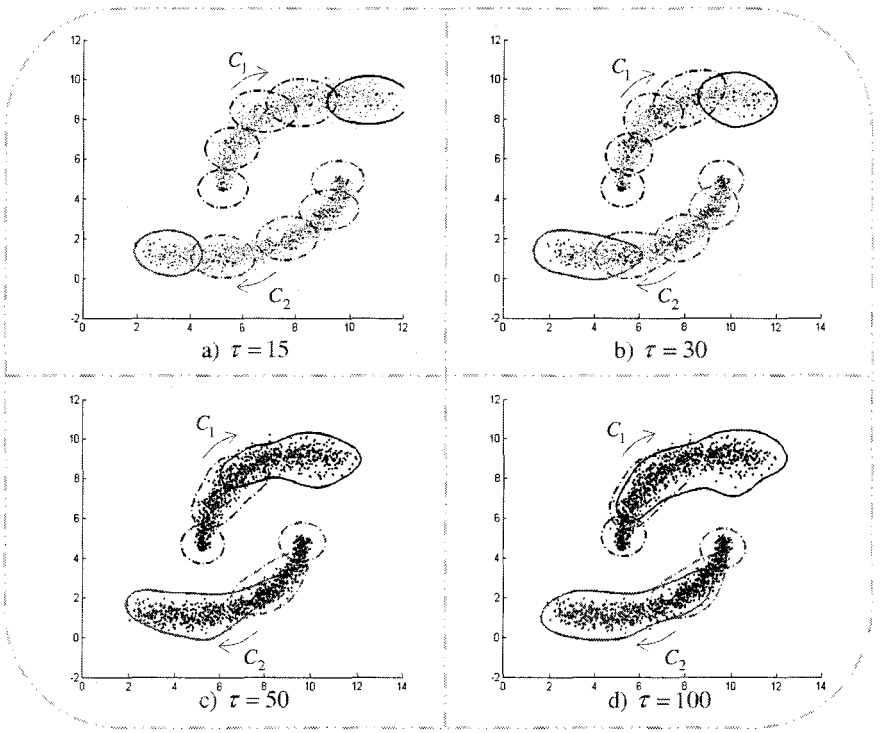

Figure 5.2.7 : Influence de τ sur le modèle de classification du SAKM. Paramètres fixés : $\lambda = 0{,}8$; $\nu = 0{,}3$; $\eta = 0{,}1$; $N_{\min} = 10$; $N_{amb} = 10$; $T = 333$.

Paramètre τ	**15**	**30**	**50**	**100**
Tps d'exécution (sec.)	4,55	4,68	5,38	6,21
Nb classes (SVs)	2(30)	2(60)	2(100)	2(148)

Tableau 5.2.7 : Indicateurs de performance du SAKM en fonction du paramètre τ.

Interprétation des résultats : Lorsque le paramètre τ prend des valeurs petites, le modèle de classe est défini sur un petit nombre de vecteurs support (figure 5.2.7.a). Dans ce cas, les données sont vite élaguées de l'apprentissage car les poids α_i décroissent rapidement. En revanche, pour des valeurs trop grandes de τ, il y a moins de retrait d'informations et par conséquent le modèle de classe est plus étendu (figure 5.2.7.d). Dans le tableau 5.2.7, on constate une légère augmentation du temps d'exécution du SAKM avec le paramètre τ.

On remarque que le SAKM offre une meilleure modélisation de classes évolutives que l'AUDyC, notamment pour les classes définies sur une fenêtre de taille importante (figures 5.2.4.d et 5.2.7.d). En effet, grâce à la flexibilité des fonctions d'apprentissage à noyau, le SAKM est moins exposé aux problèmes d'optima locaux.

Le paramètre τ est soumis à des conditions de réglage similaires à celles du paramètre N_P (taille de la fenêtre de définition de prototypes de l'AUDyC). Ce paramètre est réglé de façon empirique quand il n'est pas imposé par le problème à traiter.

5.2.2.4 Sensibilité du coefficient ν

Le coefficient ν n'est utilisé que dans la fonction objectif du SAKM. Il n'a donc aucune influence sur le modèle de classification du SAKM. Par contre, il est utile pour le calcul d'erreurs dans la fonction objectif du SAKM (chapitre 4, équations 4.31 et 4.32). Dans toutes les simulations, ce coefficient est fixé $\nu = 0,3$.

A partir des expériences effectuées dans cette partie, on constate que seul le réglage de deux paramètres de modèle pose quelques difficultés à chaque algorithme. Pour l'AUDyC, il s'agit du paramètre σ_{ini} de la matrice de covariance initiale et la taille N_P de la fenêtre de définition des prototypes. Quant au SAKM, ce sont le paramètre λ du noyau gaussien et le paramètre τ imposant le nombre maximal de vecteurs support définissant chaque classe. Les réglages de σ_{ini} et de λ dépendent de la distribution des données. Dans le cas évolutif, les réglages de N_P et τ nécessitent quelques fois des connaissances supplémentaires sur la taille des classes, sinon leur choix se fait de façon empirique. Pour le cas statique, il convient de fixer $N_P = \infty$ (adaptation sans oubli) pour l'AUDyC et de choisir τ pour avoir un nombre suffisant de vecteurs support pour modéliser chaque classe du SAKM.

5.3 Simulation des procédures d'apprentissage de l'AUDyC et du SAKM

Après étude du réglage des paramètres des classifieurs, nous allons tester leurs procédures d'apprentissage. Rappelons que l'originalité des algorithmes AUDyC et SAKM concerne leurs capacités d'apprentissage et d'auto-adaptation en environnement de données non-stationnaires. Chacun de ces algorithmes est élaboré avec quatre procédures d'apprentissage : création, adaptation, fusion, et évaluation. Afin de montrer les capacités d'auto-adaptation des algorithmes, des expériences sont réalisées sur des données non-stationnaires. Celles-ci sont créées de façon à décrire diverses situations qui relèvent de la problématique de la classification dynamique.

La première section de cette partie traite des situations d'apparition de classes. La seconde section s'intéresse au suivi de classes évolutives. Les expériences menées dans la

troisième et la quatrième section concernent les problèmes de fusion et scission de classes. Enfin, dans la dernière section, la robustesse de chaque algorithme est appréciée en environnement bruité. Pour toutes les expériences effectuées, les résultats de l'AUDyC et du SAKM sont présentés puis interprétés.

5.3.1 Apparition de classes

Deux grandes catégories d'apparition de classes sont intéressantes à étudier : l'apparition de classes après un changement brusque et l'apparition de classes après une dérive rapide. On propose de tester l'AUDyC et le SAKM sur plusieurs cas d'apparition de classes.

Cas 1 : Apparition d'une classe après un changement brusque

Cette expérience est réalisée avec 300 données partagées entre deux classes distinctes. Ces données sont réparties de telle sorte que les 150 premières forment la classe 1 et les 150 dernières forment la classe 2. L'apprentissage est effectué séquentiellement en passant les données les unes après les autres.

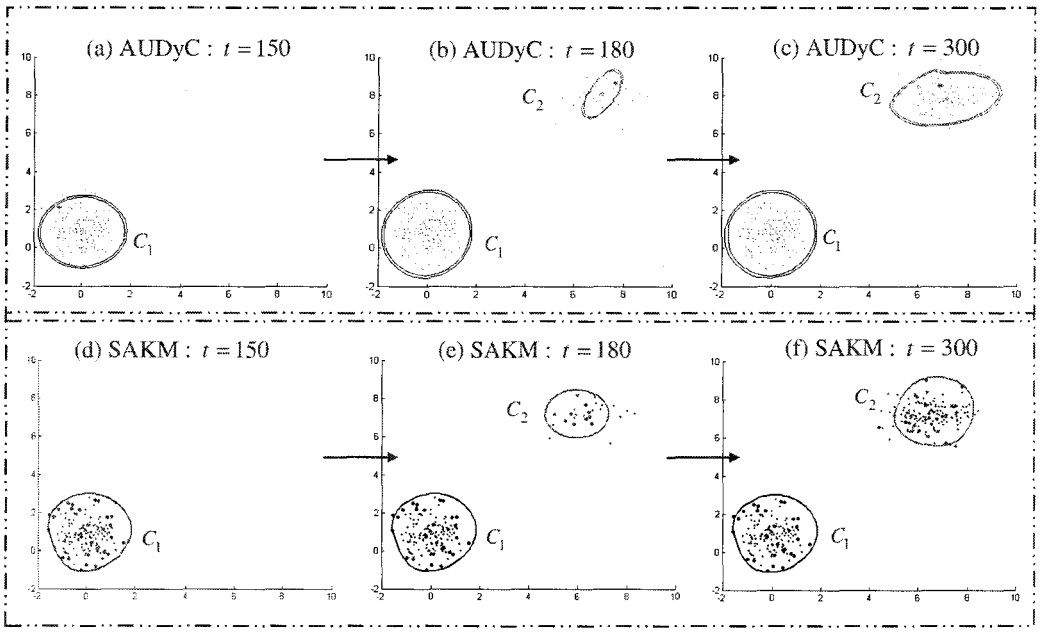

Figure 5.3.1 : Création de classe après un changement brusque. Réglage des paramètres / AUDyC : $\sigma_{ini} = 0{,}19$; $N_P = \infty$; $N_{\min} = 10$ / SAKM : $\lambda = 0{,}8$; $\eta = 0{,}1$; $\tau = 30$; $N_{\min} = 10$.

Interprétation des résultats : Les résultats de cette expérience sont présentés sur la figure 5.3.1. Chacun des deux algorithmes AUDyC et SAKM crée d'abord la première classe à l'arrivée des premières données (figures 5.3.1.a et 5.3.1.d). Cette classe est ensuite adaptée au fur et à mesure que les données arrivent. Après la 150^e donnée, il se produit un saut brusque de données dans une autre région de l'espace. A cette occasion, une nouvelle classe apparaît. L'AUDyC crée alors un prototype pour initialiser cette classe (figure 5.3.1.b) tandis le SAKM initialise son modèle en créant des vecteurs support (figure 5.3.1.e). La nouvelle

classe créée n'est considérée comme représentative que si sa cardinalité dépasse le seuil $N_{\min}$ (figures 5.3.1.c et 5.3.1.f).

Cas 2 : Apparition quasi-simultanée de deux classes après un changement brusque

Dans le présent test, on se propose d'analyser les capacités d'apprentissage de l'AUDyC et du SAKM dans un problème d'apparition quasi-simultanée de deux classes à la suite d'un saut brusque. L'expérience est réalisée avec 3 classes dont la première contient 150 données et les deux autres classes sont constituées de 125 données chacune. Les données de ces deux dernières sont acquises alternativement dans deux régions distinctes de l'espace.

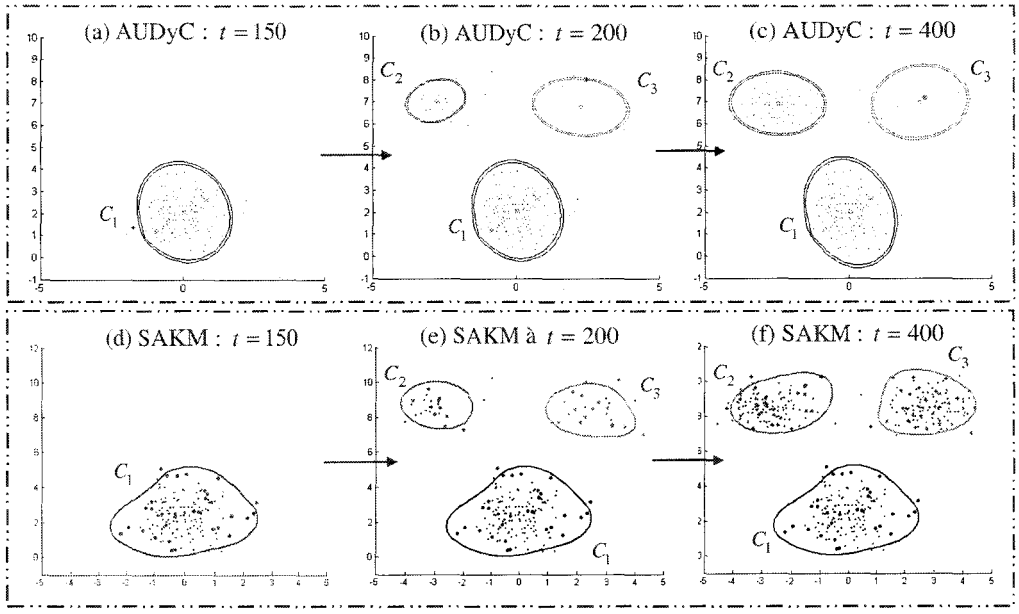

Figure 5.3.2 : Création quasi-simultanée de 2 classes après un changement brusque. Réglage des paramètres / AUDyC : $\sigma_{ini} = 0,25$; $N_P = \infty$; $N_{\min} = 10$ / SAKM : $\lambda = 0,8$; $\eta = 0,1$; $\tau = 30$; $N_{\min} = 10$.

La figure 5.3.2 illustrent les résultats produits par l'AUDyC et le SAKM au cours de cette expérience. Tout d'abord, la première classe est créée puis adaptée (figures 5.3.2.a et 5.3.2.d). Dès que leurs données se présentent, les 2 autres classes sont créées quasi-simultanément (figures 5.3.2.b et 5.3.2.e). Ces classes sont mises à jour itérativement au fur et à mesure que les données se présentent (figures 5.3.2.c et 5.3.2.f). Ces résultats montrent les aptitudes de l'AUDyC et du SAKM à créer deux ou plusieurs classes en même temps.

Cas 3 : Apparition d'une classe à données périodiques

Considérons un problème de classification dynamique dans lequel des données sont acquises de façon périodique dans la même région de l'espace. Ces données, si leur nombre est suffisant, sont révélatrices de l'apparition d'une nouvelle classe. Pour illustrer ce problème, 550 données sont générées pour former deux classes : la classe 1 avec 500 données et la classe 2 avec 50 données périodiques. Ces données sont rangées de telle sorte que chaque donnée de la classe 2 arrive après l'acquisition de 10 données de la classe 1.

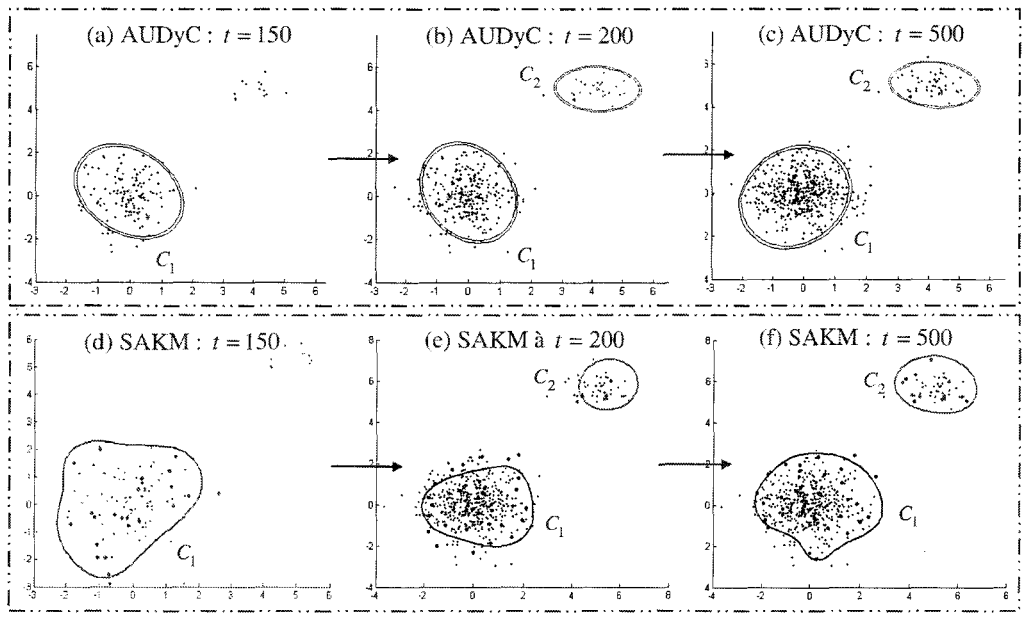

Figure 5.3.3 : Création de classe à données périodiques en utilisant l'algorithme SAKM. Réglage des paramètres /AUDyC : $\sigma_{ini} = 0,21$; $N_P = \infty$; $N_{\min} = 10$ /SAKM : $\lambda = 0,8$; $\eta = 0,1$; $\tau = 30$; $N_{\min} = 10$.

Sur les figures 5.3.3.a et 5.3.3.d, l'AUDyC et le SAKM modélisent d'abord la classe 1. Ensuite, lorsque suffisamment de données de la classe 2 sont acquises, le modèle de cette classe est calculé par l'AUDyC (figure 5.3.3.b) et le SAKM (figure 5.3.3.e). Au fur et à mesure des acquisitions, ces deux classes sont mises à jour (figures 5.3.3.c et 5.3.3.f). Les situations de création de classes à données périodiques sont peu fréquentes dans les problèmes réels, mais lorsqu'elles ont lieu, elles présentent un risque élevé pour la classification. En effet, à cause de sa lente formation, une classe à données périodiques peut être confondue avec les classes parasites et être éliminée lors de la procédure d'évaluation. La solution à ce problème passe quelquefois par le choix d'une durée T élevée entre deux procédures d'élimination et d'un seuil de cardinalité $N_{\min}$ faible. Cependant, ce réglage se fait au détriment de la robustesse des classifieurs dynamiques.

Cas 4 : Apparition de classe après une dérive rapide de données

Cette fois-ci nous analysons le comportement des deux algorithmes dans les situations où l'apparition de la nouvelle classe est précédée d'une dérive rapide de données. A cet effet, deux classes sont créées avec 150 données chacune. Dans une première expérience, une dérive rapide constituée de 10 données fait la transition entre la classe 1 et la classe 2. Dans la seconde expérience, le nombre de données de la dérive est passé à 30.

La figure 5.3.4 montre les résultats produits par l'AUDyC et le SAKM pour la première expérience. Après la création de la classe 1 (figures 5.3.4.a et 5.3.4.d), on observe une dérive rapide de données. Après la stabilisation, la classe 2 est créée (figures 5.3.4.b et 5.3.4.e). Les figures 5.3.4.c et 5.3.4.f montrent les modèles des classes estimés par l'AUDyC et le SAKM.

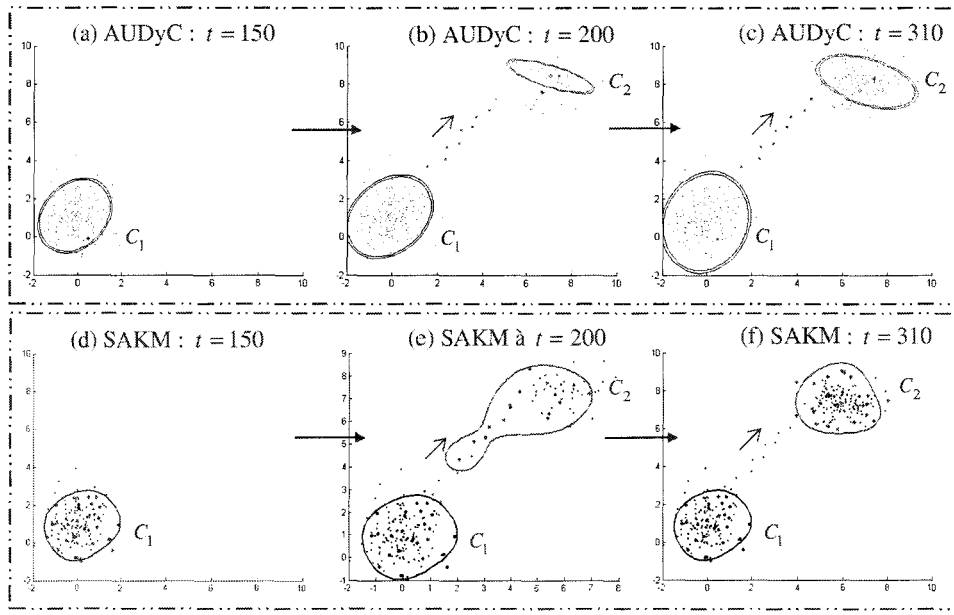

Figure 5.3.4 : Création de classe après une dérive (très) rapide de données. Réglage des paramètres / AUDyC : $\sigma_{ini} = 0{,}17$; $N_P = \infty$; $N_{\min} = 10$ / SAKM : $\lambda = 1$; $\eta = 0{,}1$; $\tau = 30$; $N_{\min} = 10$.

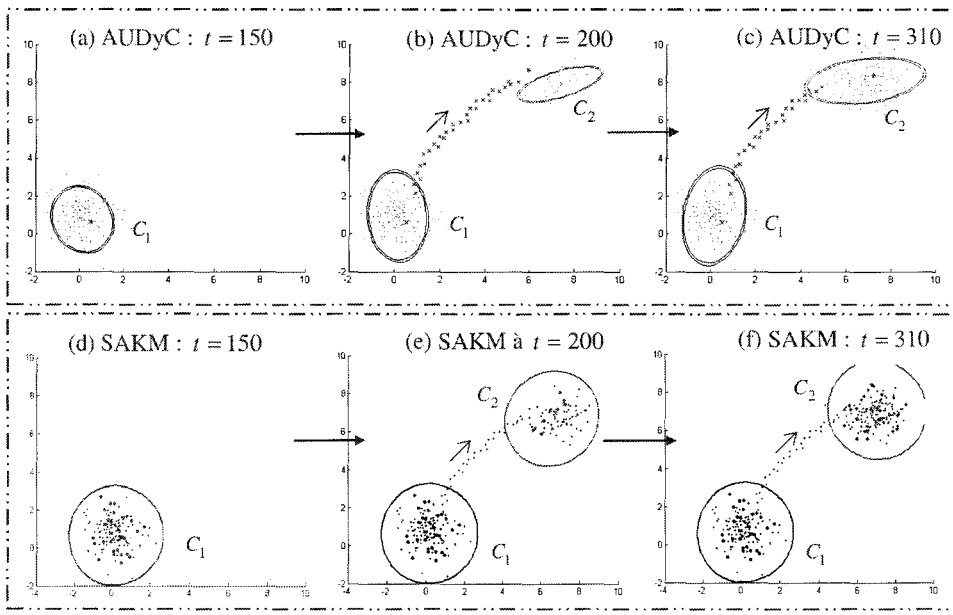

Figure 5.3.5 : Création de classe après une dérive rapide de données. Réglage des paramètres / AUDyC : $\sigma_{ini} = 0{,}17$; $N_P = \infty$; $N_{\min} = 15$ / SAKM : $\lambda = 2$; $\eta = 0{,}1$; $\tau = 30$; $N_{\min} = 15$.

En effectuant la même simulation avec plus de données de transition, on obtient sur la figure 5.3.5 des résultats proches de ceux de la figure 5.3.4. Cependant, la réalisation de cette expérience dans les meilleures conditions a nécessité d'augmenter les valeurs du seuil $N_{\min}$ afin d'éviter la création de classes sur la dérive (avant la stabilisation de données). Dans le cas de l'apprentissage du SAKM, il est aussi nécessaire d'augmenter la valeur du paramètre λ afin d'éviter d'affecter les données de la dérive à la classe 1.

5.3.2 Suivi de classes évolutives

Dans cette section, on teste les capacités de nos classifieurs dynamiques dans le problème de suivi d'évolution. Trois expériences sont réalisées : dans l'expérience 1, des données non-stationnaires sont créées pour décrire 2 classes évolutives. Les données des expériences 2 et 3 sont créées de façon similaire mais décrivent respectivement 3 et 4 classes évolutives. Dans les trois expériences, 1000 données non-stationnaires sont utilisées sur chaque trajectoire. L'évolution de chaque classe se fait par glissement progressif de la moyenne et l'augmentation du volume de la classe.

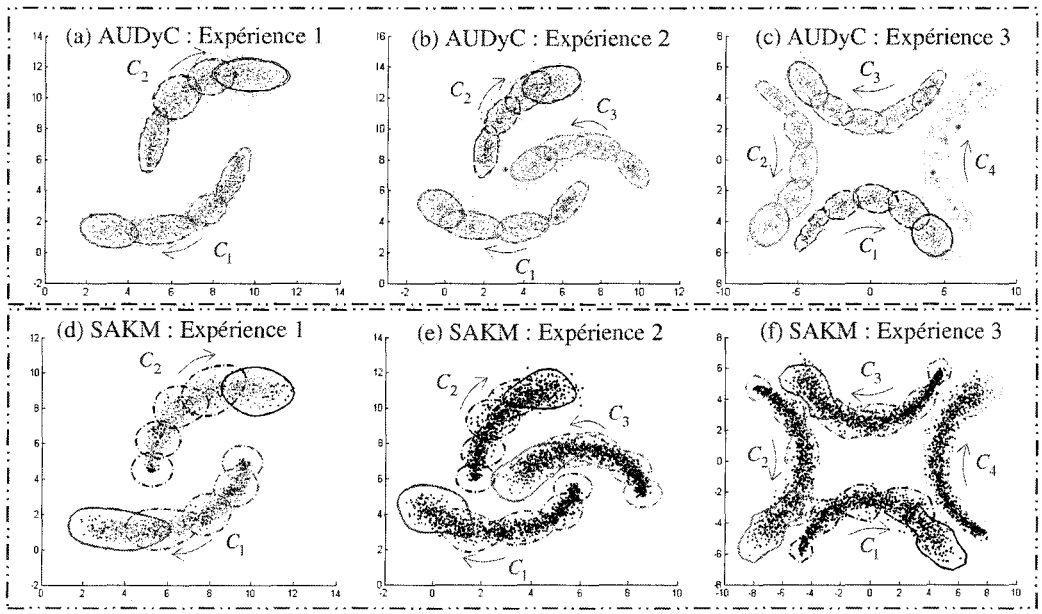

Figure 5.3.6 : Suivi d'évolution de classes avec les classifieurs dynamiques. et SAKM. Paramètres /AUDyC : $\sigma_{ini} = 0{,}55$; $N_P = 110$; $N_{\min} = 10$ / SAKM : $\lambda = 0{,}65$; $\eta = 0{,}1$; $\tau = 30$; $N_{\min} = 10$.

Interprétation des résultats : la figure 5.3.6 montre les capacités de suivi d'évolution des algorithmes. L'AUDyC modélise les classes évolutives avec des prototypes gaussiens. Le suivi d'évolution de classes est obtenu grâce à la procédure d'adaptation élaborée avec des règles d'ajout et de retrait d'informations (figures 5.3.6.a, b, c). Le SAKM a de bonnes aptitudes à effectuer le suivi de classes évolutives de contours plus ou moins complexes sans *a priori* sur la distribution des données (figures 5.3.6.d, e, f). La modélisation dynamique de classes est obtenue à partir des règles de mise à jour itératives basées sur le gradient.

5.3.3 Fusion de classes

La fusion de classes est un mécanisme dynamique dont le traitement pose des difficultés quelquefois complexes. Afin d'analyser les capacités des classifieurs dynamiques, trois expériences de fusion sont réalisées sur des données de simulation. 2000 données non-stationnaires sont créées pour décrire 2 classes évolutives qui fusionnent. Pour chacune des expériences, on présente les résultats de l'AUDyC et ceux du SAKM. Afin de montrer les

améliorations apportées par la nouvelle procédure de fusion, les résultats donnés par l'ancienne version de l'AUDyC (AUDyC v1) sont également présentés. Dans l'ancienne version de l'AUDyC, l'ambiguïté est traitée en fonction du réglage, soit par l'adaptation des prototypes gagnants ou soit par leur fusion.

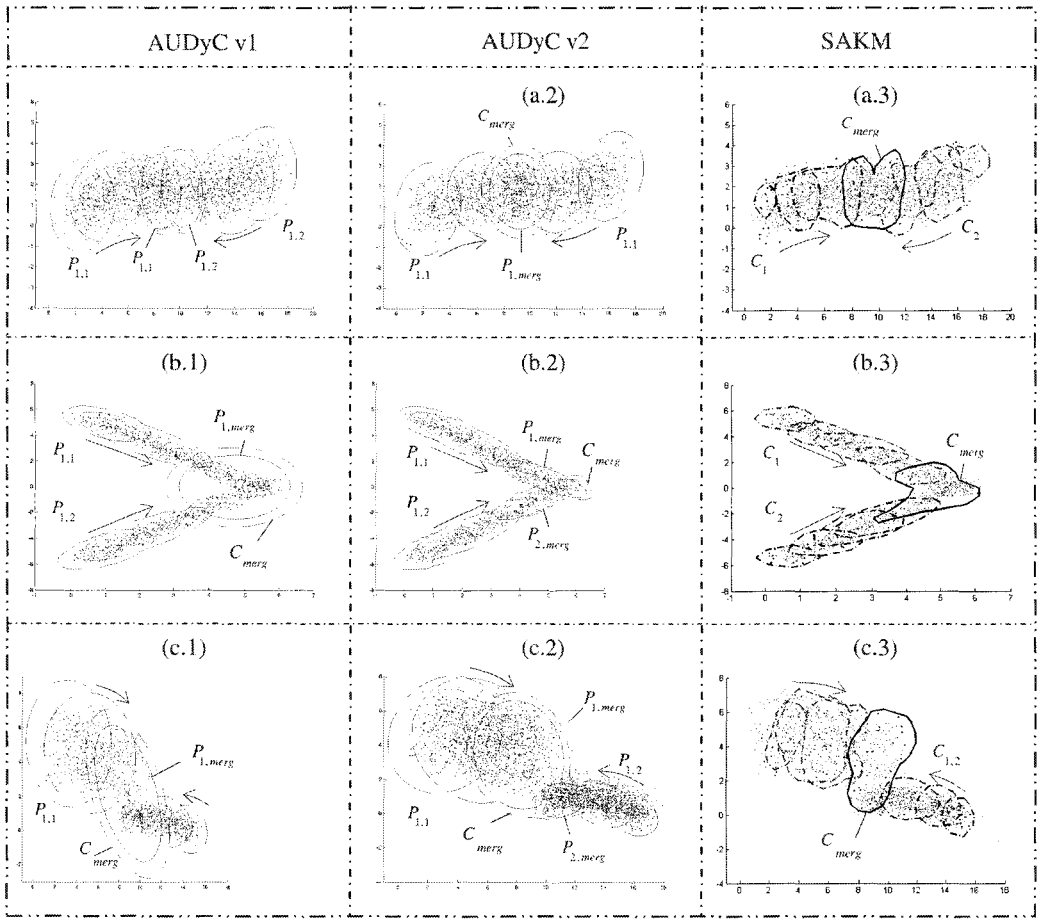

Figure 5.3.7 : Fusion de classes en utilisant les classifieurs dynamiques : AUDyC v1 (ancienne version), AUDyC v2 (nouvelle version) et SAKM. Réglage de paramètres / AUDyC : $\sigma_{ini} = 0.5$; $N_P = 100$; $N_{\min} = 10$; $N_{amb} = 5$ /SAKM : $\lambda = 0.65$; $\eta = 0{,}1$; $\tau = 30$; $N_{\min} = 5$; $N_{amb} = 5$.

Interprétation des résultats : Si l'algorithme AUDyC v1 est réglé pour adapter les prototypes gagnants, il y a un risque important de chevauchement de prototypes (figure 5.3.7.a.1). En revanche, si l'option de fusion est choisie, les problèmes de sous-apprentissage deviennent fréquents (figures 5.3.7.b.1 et 5.3.7.c.1). La nouvelle version AUDyC v2 est dotée d'une procédure capable de calculer le modèle de classe en décidant de façon autonome d'adapter ou de fusionner les prototypes. De cette façon, on évite les défauts de modélisation constatés dans la première version (figures 5.3.7.a.2, 5.3.7.b.2 et 5.3.7.c.2). Les résultats des mêmes expériences effectuées avec le SAKM sont présentés sur les figures 5.3.7.a.3, 5.3.7.b.3 et 5.3.7.c.3. Grâce à ses capacités de modélisation de classes complexes, le SAKM détermine des modèles de fusion de classes en évitant les problèmes d'optima locaux.

5.3.4 Scission de classes

Dans cette section, nous allons étudier le comportement des algorithmes AUDyC et SAKM dans un problème de scission de classes. Rappelons que contrairement à l'AUDyC, le SAKM n'est pas doté d'une procédure de scission (chapitre 4, section 4.3.2.5). L'expérience est réalisée avec 2000 données non-stationnaires décrivant la scission d'une classe en 2 nouvelles classes évolutives.

Interprétation des résultats : la figure 5.3.8.a montre que l'AUDyC détecte la scission de classe et estime correctement les modèles de deux nouvelles classes issues de la scission. Quant au SAKM, il ne détecte pas la séparation des données et met à jour continuellement la classe en adaptant son modèle au fur et à mesure de l'arrivée des données. Le modèle est ainsi estimé sur deux groupes de données évoluant suivant deux trajectoires distinctes. De ce fait, on obtient un modèle constitué de deux contours disjoints représentant la même classe (figure 5.3.8.b). Cette description du modèle ne satisfait pas à la contrainte de continuité de classes recherchée en classification de données spatiales. Le SAKM ne convient donc pas pour résoudre les problèmes de scission de classes. Pour pallier cet inconvénient, la mise en œuvre de quelques règles supplémentaires est nécessaire. Les réflexions menées dans ce sens n'ont jusqu'à présent pas permis d'élaborer une procédure de scission fiable pour le SAKM.

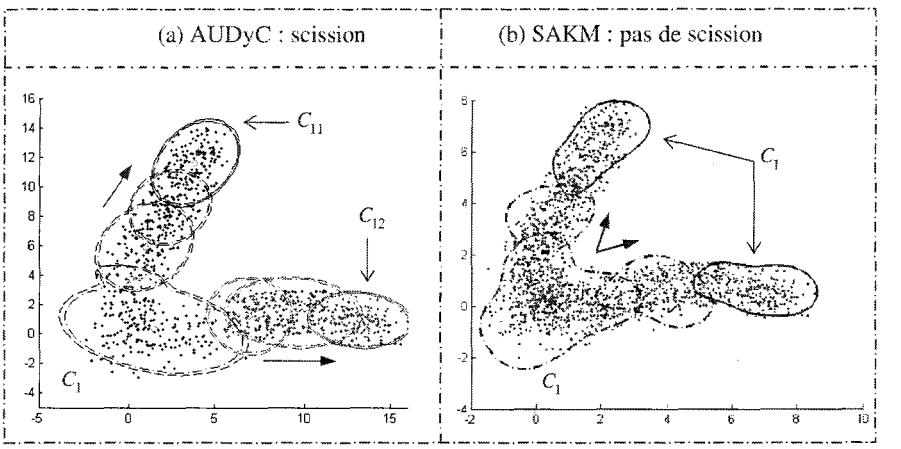

Figure 5.3.8 : Scission de classe avec l'AUDyC. (b) Pas de scission avec le SAKM. Paramètres /AUDyC: $\sigma_{ini} = 0{,}5$; $N_P = 100$ /SAKM: $\lambda = 0{,}80$; $\eta = 0{,}1$; $\tau = 30$; / $N_{\min} = 10$; $N_{amb} = 5$.

5.3.5 Élimination de classes parasites (bruit)

Les tests menés dans cette section visent à montrer le rôle considérable joué par le mécanisme d'élimination dans la robustesse des algorithmes face au bruit. En effet, le bruit occasionne la création d'un nombre important de classes parasites affectant la qualité de classification. Trois expériences sont menées pour illustrer l'importance et les capacités du mécanisme d'élimination. L'expérience 1 et l'expérience 2 sont effectuées avec 3 classes statiques constituées chacune de 150 données. Ces données sont mixées avec un bruit uniforme (13 x 13 données bruit) dans l'expérience 1, et avec un bruit gaussien (100 données bruit) dans l'expérience 2. Enfin, l'expérience 3 est effectuée avec 2 classes évolutives

constituées chacune de 1000 données. Sur la trajectoire de chaque classe, un bruit gaussien dynamique constitué de 200 données suit l'évolution des classes.

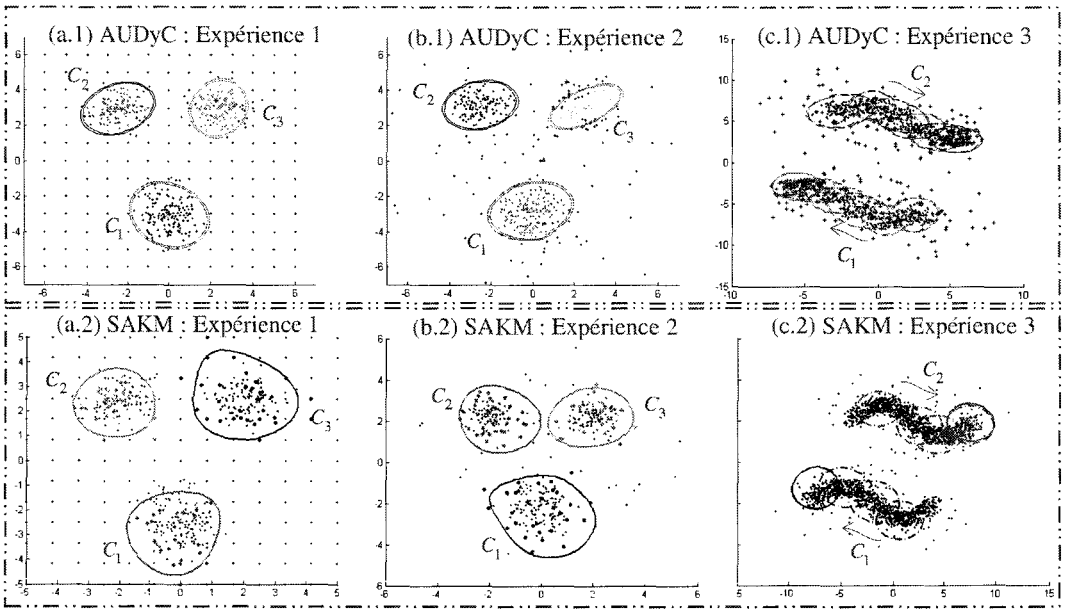

Figure 5.3.9 : Classification dynamique en environnement bruité avec l'AUDyC et le SAKM. Paramètres AUDyC : $N_{\min} = 30$; $N_{amb} = 5$; $T = 200$. Expériences 1 et 2 (classes statiques) : $\sigma_{ini} = 0,2$, $\lambda = 0,65$, $N_P = \inf$, Expérience 3 (classes évolutives) : $\sigma_{ini} = 1$, $\lambda = 1,8$, $N_P = 50$.

Interprétation des résultats : Les figures 5.3.9.a.1 et 5.3.9.b.1 montrent les bonnes aptitudes de l'AUDyC à effectuer l'apprentissage de classes statiques en présence d'un bruit uniforme ou d'un bruit gaussien. De même, la modélisation adaptative de classes évolutives est effectuée convenablement malgré le bruit gaussien dynamique (figure 5.3.9.c.1). Les résultats produits par le SAKM sont moins convaincants que ceux de l'AUDyC en terme de robustesse au bruit. L'algorithme retrouve les trois classes statiques sur les figures 5.3.9.a.2 et 5.3.9.b.2 et effectue le suivi de deux classes évolutives dans l'expérience de la figure 5.3.9.c.2. Cependant, l'estimation des modèles est affectée par le bruit aux alentours de chaque classe. Cette sensibilité au bruit est due à l'instabilité du modèle adapté itérativement en utilisant la méthode du gradient. En revanche, en environnement bruité, nous avons constaté que la rapidité du SAKM est peu affectée contrairement à celle de l'AUDyC. En réalité, le temps d'exécution de l'AUDyC augmente à cause de la quantité des données rejetées qui sont re-introduites à plusieurs reprises dans le processus d'apprentissage. Par ailleurs, si le paramètre $N_{\min}$ est choisi trop grand, il y a un risque d'élimination de classes représentatives. Par contre, si ce paramètre est choisi trop faible, la qualité de modélisation est susceptible d'être affectée par le bruit.

Les expériences menées dans cette partie ont montré les capacités d'auto-adaptation des deux classifieurs dynamiques. Ces algorithmes adaptent leur modèle de classification face aux phénomènes d'apparition de classes grâce à leurs procédures de création de classes. D'après

les résultats de plusieurs tests, l'apparition de classes après un saut brusque, ou celle précédée d'une dérive rapide sont correctement traitées par l'AUDyC et le SAKM. Lorsque les classes évoluent dans le temps, leur mise à jour est effectuée de façon à suivre ces évolutions. Dans les problèmes de suivi de classes complexes, les modèles successifs produits par le SAKM sont meilleurs que ceux estimés par l'AUDyC. En effet, l'AUDyC est victime de problèmes de sous-apprentissage à cause de l'utilisation du modèle gaussien (modèle paramétrique). En revanche, plusieurs expériences ont montré que l'AUDyC offre des résultats très intéressants dans les problèmes de suivi de classes gaussiennes. La fusion de classes est également un phénomène pris en compte par les deux algorithmes. Quant à la scission de classes, seul l'AUDyC dispose d'une procédure adaptée à ce phénomène dynamique. Enfin, en environnement bruité, moyennant un réglage adéquat de certains paramètres, on obtient une qualité de modélisation plus ou moins bonne. On note que le modèle de classification du SAKM est plus affecté par le bruit que celui de l'AUDyC.

5.4 Complexité des algorithmes AUDyC et SAKM

Dans le chapitre 4 (section 4.2.5.2 et 4.3.5.2), les complexités algorithmiques de l'AUDyC et du SAKM ont été déterminées par le calcul du nombre d'opérations. A l'issue de ce calcul, la complexité de l'AUDyC est évaluée à $O(D^3 \times J^2 \times L)$ tandis que celle du SAKM vaut $O(D \times \tau^2 \times M \times L)$. En réalité, ces grandeurs de complexité algorithmique sont estimées dans les conditions les plus défavorables. Ces estimations sont effectuées avec l'hypothèse pessimiste qu'à chaque acquisition, toutes les procédures d'apprentissage sont utilisées pour traiter la donnée. Ces complexités sont donc surévaluées et représentent en réalité, les bornes supérieures des temps d'exécution. En effet, dans la plupart des cas, seuls le critère de similarité et la procédure d'adaptation sont utilisés à chaque acquisition. La fusion et la scission de classes qui sont des procédures gourmandes ne sont qu'occasionnellement appelées. En faisant abstraction de ces procédures, la complexité de l'AUDyC se limite à $O(D^3 \times J \times L)$ et celle du SAKM à $O(D \times \tau \times M \times L)$.

Les tests menés dans cette partie ont pour but d'analyser le temps d'exécution de chaque algorithme d'un point de vue expérimentale[2] puis de comparer le résultat avec la complexité estimée par le calcul. Dans la section 5.4.1, on présente une simulation montrant l'influence de la dimension D de données sur la complexité des deux algorithmes. Ensuite, la section 5.4.2 met en évidence d'une part, l'influence du nombre J de prototypes sur le temps d'exécution de l'AUDyC et d'autre part, l'influence du nombre $\tau \times M$ de vecteurs support sur le temps d'exécution du SAKM. Enfin dans la section 5.4.3, la rapidité de chacun des algorithmes est analysée en fonction du nombre L de données à traiter.

[2] Toutes les expériences sont effectuées avec le logiciel Matlab en utilisant un ordinateur ayant les caractéristiques suivantes : Processeur P4 - 2Ghz, RAM 256 Mo.

5.4.1 Influence de la dimension sur la complexité algorithmique

Conditions de réalisation : Cette expérience est réalisée avec 3 classes statiques de 500 données chacune. Le nombre de données à traiter est donc maintenu fixe à $L=1500$. Les données de chaque classe sont générées en utilisant une loi de densité gaussienne (écart type $\sigma_{d\in[1,..,D]}=1$ sur chaque dimension). L'algorithme AUDyC est capable de modéliser chaque classe avec un prototype gaussien de telle sorte que le nombre de prototypes soit $J=3$ (i.e $J=M$ nombre de classes). Dans le cas de l'apprentissage du SAKM, chaque classe est modélisée en utilisant 30 vecteurs support de telle sorte que le nombre total de vecteurs support soit figé à $M\times\tau=90$. Le test de complexité est ainsi réalisé en faisant varier uniquement la dimension D ; le nombre L de données et le nombre J de prototypes de l'AUDyC (respectivement le nombre $M\times\tau$ de vecteurs support du SAKM) étant maintenus fixes tout au long de l'apprentissage.

Interprétation des résultats de l'AUDyC : La figure 5.4.1.a illustre la modélisation de classes en $2D$. Dans le tableau 5.4.1, on constate que le temps d'exécution de l'AUDyC augmente avec la dimension des données. En utilisant la méthode des moindres carrés et l'hypothèse que le temps d'exécution peut s'écrire sous la forme : $T_{exe}=a_3D^3+a_2D^2+a_1D^1+a_0$ (complexité de l'ordre $O(D^3)$), on obtient les coefficients réels $a_3=-1{,}5\cdot10^{-3}$, $a_2=0{,}11$, $a_1=-1{,}13$ et $a_0=11{,}98$ (figure 5.4.1.c). A partir de ce résultat, on constate que la complexité de l'algorithme AUDyC est polynomiale. Mais, étant donné que le coefficient pondérant D^3 est négatif (i.e. T_{exe} décroît si D^3 augmente), l'ordre de complexité $O(D^3)$ est plutôt une borne supérieure de la complexité expérimentale. On trouve deux explications possibles à cela :

- L'hypothèse (pour calculer la complexité) selon laquelle l'inversion de matrice est faite par la méthode de *Gauss-Jordan* ne correspond-elle pas à la réalité. Il se peut que le logiciel *Matlab* utilise une méthode plus rapide pour effectuer ce calcul.
- Le nombre de points utilisés pour calculer les cœfficients a_i sont insuffisants pour approximer la courbe exacte. Dans ce cas, il est nécessaire d'effectuer d'autres expériences pour des données de dimension plus grande.

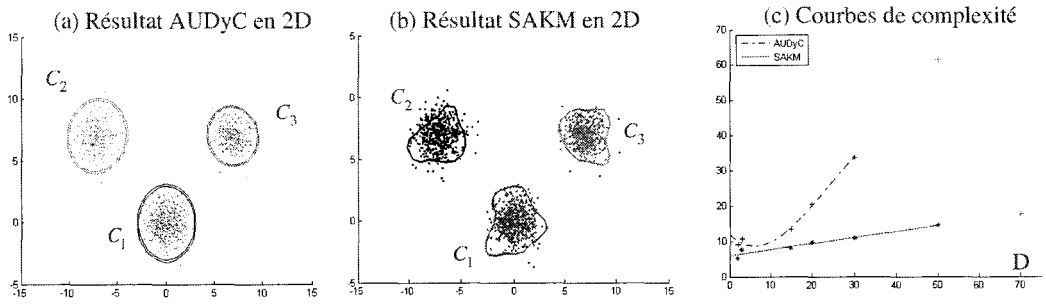

Figure 5.4.1: Influence de la dimension : Modélisation 2D des classes ($M=3$ et $L=1500$).
Paramètres /AUDyC : $N_p=\infty$ /SAKM : $\nu=0{,}3$; $\eta=0{,}1$; $\tau=50$ / $N_{\min}=20$, $N_{amb}=5$, $T=100$.

Malheureusement, lorsque la dimension devient trop élevée, il se produit une trop forte augmentation du temps d'exécution et une mauvaise (ou non) convergence qui se traduit par la chute de la vraisemblance L_{poss} (Tableau 5.4.1). L'AUDyC est en effet, confronté à un problème connu sous le nom de "fléau de la dimensionnalité" (*curse of dimensionality*) [Jain & al., 2000]. Pour les dimensions $D > 30$, le modèle de classification est mauvais et la convergence, lorsqu'elle est possible, nécessite l'augmentation de la valeur de σ_{ini}.

Dimension D	2	3	15	20	30	50
Paramètre σ_{ini}	*3*	*3*	*3*	*3*	*3*	*4*
Vraisemblance L_{poss}	-1,3	-2,05	-12,18	-16,55	-24,28	-29,60
Tps d'exécution (sec.)	**9,11**	**10,76**	**13,42**	**20,57**	**33,87**	61,44

Tableau 5.4.1 : Indicateurs de performance de l'AUDyC : Complexité algorithmique en fonction de la dimension de données.

Interprétation des résultats du SAKM : Le résultat de la modélisation du SAKM est présentée sur la figure 5.4.2.b. Dans le tableau 5.4.2, on constate que le temps d'exécution du SAKM croît avec la dimension des données. D'après le calcul, la complexité algorithmique est linéaire en D c'est-à-dire que le temps d'exécution peut s'exprimer par : $T_{exe} = a_1 D + a_0$. A l'aide de la régression linéaire, on détermine les coefficients : $a_1 = 0{,}17$ et $a_0 = 5{,}94$. La courbe de complexité du SAKM est donnée sur la figure 5.4.2.c. Ce résultat pratique est en adéquation avec la complexité théorique du SAKM, c'est-à-dire, de l'ordre de $O(D)$. Cependant, on note une dégradation de la qualité de modélisation pour des dimensions supérieures à 50, qui se traduit par une augmentation importante du risque E_{learn} et des problèmes de mauvaise convergence. L'algorithme SAKM est également victime du "fléau de la dimensionnalité".

Dimension D	2	3	15	20	30	50	70
Paramètre λ	*0,68*	*0,55*	*0,04*	*0,03*	*0,02*	*0,01*	8.10^{-3}
Risque E_{learn}	0,10	0,11	0,09	0,10	0,12	0,14	0,39
Tps d'exécution (sec.)	**5,27**	**7,68**	**8,05**	**9,68**	**11,03**	**14,55**	17,65

Tableau 5.4.2 : Indicateurs de performance du SAKM : Complexité algorithmique en fonction de la dimension de données.

Dans la littérature, très peu de travaux traitent du "fléau de la dimensionnalité" [Bellman, 1961 ; Jain & al., 2000]. L'apprentissage en grande dimension est un problème difficile. En réalité, les intuitions géométriques valables en faible dimension peuvent se révéler fausses en grande dimension. Par conséquent, certains classifieurs qui fonctionnement bien en faible dimension peuvent avoir de très pauvres performances en grande dimension. A cause de ce "fléau de la dimensionnalité", la qualité de classification de l'AUDyC n'est

satisfaisante que si la dimension des données est inférieure 30 et celle du SAKM inférieure à 50, pour les expériences que nous avons réalisées. Dans les problèmes de grandes dimensions, il est donc important de réduire au préalable, la dimension de données si cela est possible avant d'utiliser les algorithmes. Plusieurs techniques sont développées pour la réduction de dimension de données [Thiria, 1996].

D'autres tests de complexité ont été effectués sur des données décrivant des classes évolutives. Les résultats de ces tests sont similaires à ceux obtenus dans le cas de classes statiques. Pour les problèmes de grande dimension, le SAKM a des performances plus élevées que l'AUDyC en terme de complexité algorithmique.

5.4.2 Influence du nombre de prototypes sur la complexité de l'AUDyC et Influence du nombre de vecteurs support sur la complexité du SAKM

Conditions de réalisation : Dans cette expérience, la dimension et le nombre des données sont fixés respectivement à $D=2$ et $L=3000$. Plusieurs tests sont effectués en augmentant le nombre M de classes gaussiennes (classes mono-prototypes). De cette façon, on obtient une variation du nombre J des prototypes dans le cas de l'apprentissage de l'AUDyC et une variation du nombre $M \times \tau$ de vecteurs support dans le cas de l'apprentissage du SAKM. On observe ainsi l'influence de ces grandeurs sur la rapidité de nos deux algorithmes de classification dynamique d'un point de vue pratique.

Interprétation des résultats de l'AUDyC : Sur les figures 5.4.2, sont présentés les résultats de la classification de l'AUDyC pour six tests. Dans le tableau 5.4.3, on note une croissance du temps d'exécution avec le nombre de prototypes. En supposant que cette croissance est linéaire en J, on obtient par les moindres carrées la courbe de complexité : $T_{exe} = 0{,}9 \cdot J + 24$. Or, sur la figure 5.4.4, les points semblent avoir une allure logarithmique. La cause de cette différence entre la complexité théorique (linéaire en J si on fait abstraction de la fusion et de la scission) et celle pratique (logarithmique en J) réside probablement dans la difficulté de connaître avec exactitude toutes les opérations effectuées par l'algorithme au cours de l'apprentissage. Dans la pratique, des prototypes peuvent fusionner ou des classes se scinder localement avant la convergence du modèle de classification. A cause de ces opérations (imprévisibles) de complexité $O(J^2)$, le temps d'exécution de l'algorithme ne peut croître linéairement. La détermination de cette complexité avec précision a peu d'intérêt dans le cas présent d'autant plus qu'une complexité linéaire ou logarithmique est satisfaite pour l'AUDyC. Par ailleurs, la qualité de modélisation n'est pas altérée par l'augmentation du nombre de prototypes, au contraire, la vraisemblance L_{poss} croît avec J (voir Tableau 5.4.3).

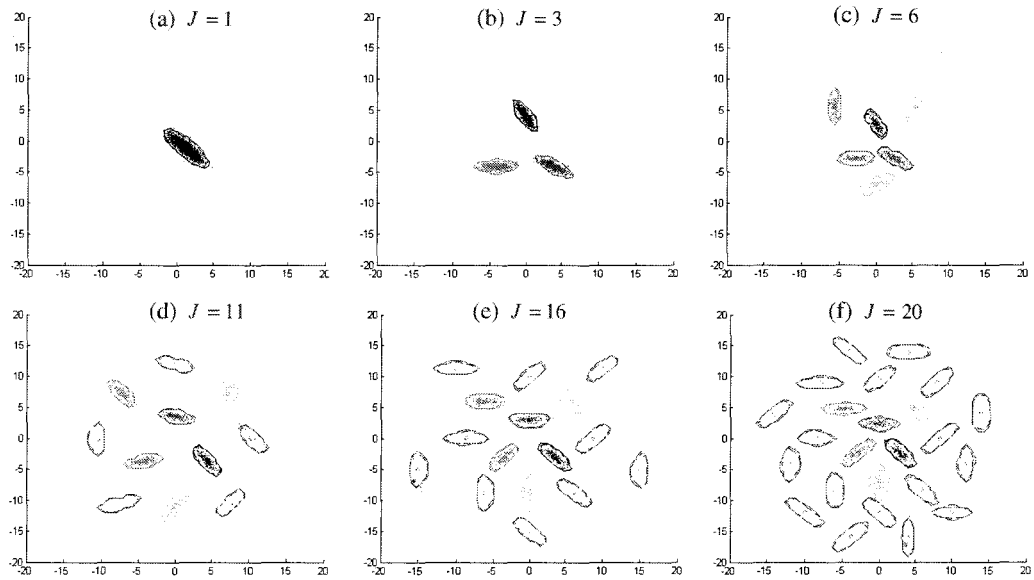

Figure 5.4.2 : Influence du nombre de prototypes : Modélisation de l'AUDyC ($L = 3000$ et $D = 2$).
Paramètres : $\sigma_{ini} = 0,6$; $N_P = \infty$; $N_{\min} = 20$; $N_{amb} = 5$; $T = 100$.

Nb de prototypes J	**1**	**3**	**6**	**11**	**16**	**24**
Vraisemblance L_{poss}	-2,97	-0,65	-0,19	-0,08	-0,09	-0,06
Tps d'exécution (sec.)	**17,59**	**29,19**	**33,03**	**37,83**	**40,23**	**42,5**

Tableau 5.4.3 : Indicateurs de performance de l'AUDyC : Complexité algorithmique en fonction du nombre de prototypes.

Interprétation des résultats du SAKM : Les figures 5.4.3 illustrent les résultats de la modélisation du SAKM. Dans le tableau 5.4.4, on constate une augmentation du temps d'exécution avec le nombre $N_{sv} = M \times \tau$ de vecteurs support. La complexité algorithmique du SAKM est théoriquement linéaire en N_{sv}. A partir des résultats obtenus, la courbe de complexité est estimée telle que : $T_{exe} = 0,05 \cdot N_{sv} + 6,5$ (figure 5.4.4). Cette courbe est cohérente avec la complexité déterminée par le calcul. Cependant, dans le tableau 5.4.4, on constate une augmentation du risque E_{learn} qui semble affirmer qu'un nombre trop important de vecteurs support dégrade la qualité de modélisation du SAKM.

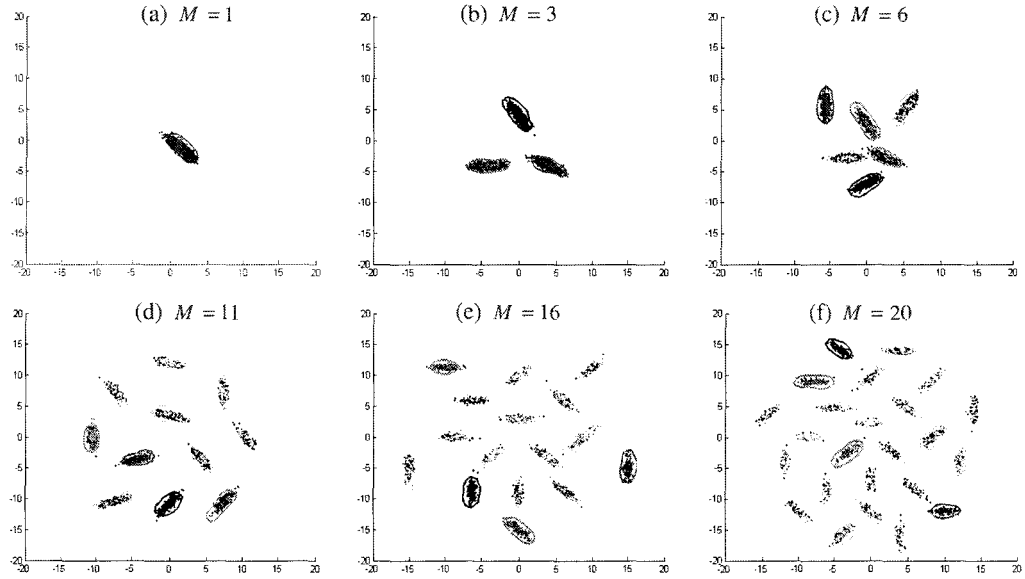

Figure 5.4.3 : Influence du nombre de prototypes : Modélisation de classes du SAKM ($L = 3000$ et $D = 2$). Paramètres : $\lambda = 0,8$; $\nu = 0,3$; $\eta = 0,1$; $\tau = 50$; $N_{\min} = 20$; $N_{amb} = 5$; $T = 100$.

Nb de SVs $M \times \tau$	**30**	**90**	**180**	**330**	**480**	**712**
Risque E_{learn}	0,04	0,16	0,28	0,43	0,48	0,50
Tps d'exécution (sec.)	**7,71**	**10,58**	**15,51**	**22,90**	**30,46**	**40,47**

Tableau 5.4.4 : Indicateurs de performance du SAKM : Complexité algorithmique en fonction du nombre de vecteurs support.

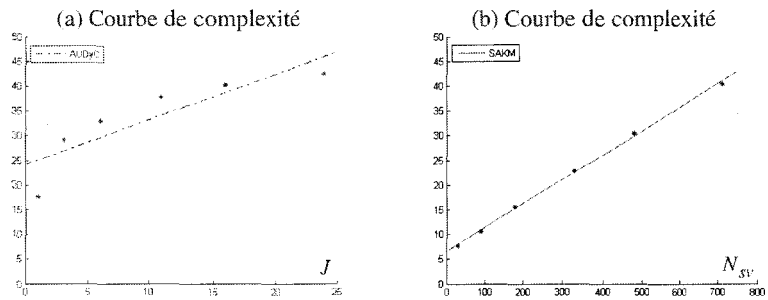

Figure 5.4.4 : Évolution du temps d'exécution de l'AUDyC en fonction du nombre J de prototypes et celle du SAKM en fonction du nombre N_{sv} de vecteurs support

5.4.3 Influence du nombre de données sur la complexité algorithmique

Conditions de réalisation : L'intérêt de ce test est d'évaluer la rapidité de nos algorithmes dans les applications hors-ligne. L'objectif consiste à analyser la complexité en fonction de la quantité de données de la base d'apprentissage. Pour réaliser cette expérience, la dimension de données et le nombre de classes sont maintenus fixes : $D = 2$ et $M = J = 3$. Plusieurs tests sont effectués en variant le nombre L de données.

Interprétation des résultats de l'AUDyC : Rappelons que le calcul du nombre d'opérations de l'AUDyC a montré que la complexité algorithmique est linéaire en fonction de nombre L de données. Les résultats du tableau 5.4.5 présente les temps d'exécution de l'AUDyC pour plusieurs valeurs du nombre L. A partir de ces résultats, la courbe de complexité est estimée : $T_{exe} = 13 \cdot 10^{-3} L - 0{,}52$ (figure 5.4.5.c). Ce résultat pratique est bien cohérent avec celui trouvé par le calcul.

Interprétation des résultats du SAKM : Le calcul de complexité effectué pour le SAKM a montré que le temps d'exécution croît linéairement avec le nombre L de données. Les expériences menées dans cette section confirment ce résultat. En effet, à partir du tableau 5.4.6 la courbe de complexité est : $T_{exe} = 6 \cdot 10^{-3} L + 0{,}9$ (figure 5.4.5.c). Par ailleurs, on note une décroissance du risque E_{learn} indiquant une amélioration de la qualité de classification lorsque le nombre de données d'apprentissage augmente (i.e. apport d'informations supplémentaires).

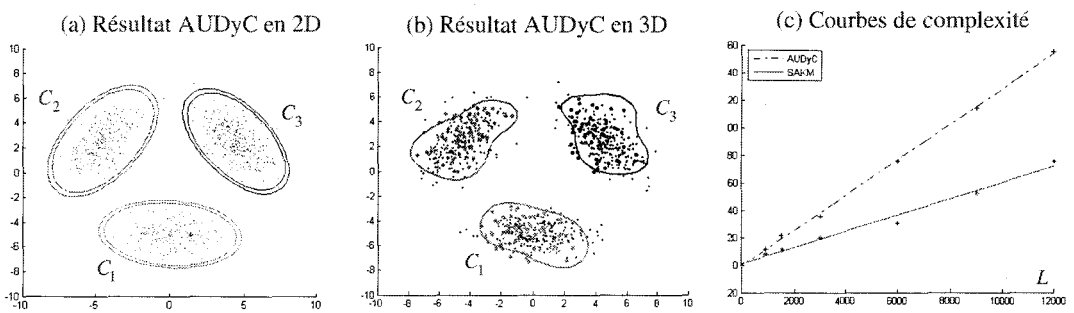

Figure 5.4.5: Influence du nombre de données : Modélisation 2D de classes ($L = 3000$ et $D = 2$). Paramètres /AUDyC : $\sigma_{ini} = 0{,}8$ $N_P = \infty$ /SAKM : $\lambda = 1; \eta = 0{,}1;$ $\tau = 50$ / $N_{\min} = 20;$ $N_{amb} = 5$.

Nb de données L	**900**	**1500**	**3000**	**6000**	**9000**	**12000**
Vraisemblance L_{pass}	-1,24	-1,98	-1,59	-1,70	-1,53	-1,50
Tps d'exécution (sec.)	**11,23**	**21,51**	**35,53**	**75,39**	**113,76**	**155,07**

Tableau 5.4.5 : Indicateurs de performance de l'AUDyC : Complexité algorithmique en fonction du nombre de données.

Nb de données L	**900**	**1500**	**3000**	**6000**	**9000**	**12000**
Risque E_{learn}	0,49	0,25	0,28	0,15	0,14	0,12
Tps d'exécution (sec.)	**7,81**	**10,92**	**19,77**	**30,98**	**52,32**	**75,64**

Tableau 5.4.6 : Indicateurs de performance du SAKM : Complexité algorithmique en fonction du nombre de données.

Les résultats des expériences réalisées dans la présente partie ont montré que pour chacun des nos deux classifieurs, la complexité expérimentale est au pire égale à celle estimée par le calcul. Or, ces complexités polynomiales d'ordre $O(D^3 \times J^2 \times L)$ pour l'AUDyC et d'ordre $O(D \times \tau^2 \times M \times L))$ pour le SAKM sont satisfaisantes pour la plupart des problèmes. Il est cependant plus intéressant de travailler en dimension faible pour assurer des bonnes performances des algorithmes et éviter le "fléau de la dimensionnalité. Enfin, comparativement à l'AUDyC, l'algorithme SAKM est plus rapide. Ceci est prouvé par le calcul théorique et par les résultats expérimentaux.

5.5 Synthèse pratique : Analyse comparée de l'AUDyC et du SAKM

Les expériences effectuées dans ce chapitre ont montré, d'un point de vue pratique, les capacités et performances de deux algorithmes AUDyC et SAKM. A travers l'analyse des résultats de ces expériences, on propose une synthèse pratique visant d'une part, à résumer les capacités de chaque algorithme et d'autre part, à comparer leurs performances.

- **Réglage des paramètres**

Dans la mise en œuvre des classifieurs dynamiques AUDyC et SAKM, un certain nombre de paramètres sont utilisés, la plupart pour des besoins de flexibilité et de robustesse. Les expériences menées dans la partie 5.2 ont montré que seuls quelques uns de ces paramètres sont difficiles à régler. Il s'agit des paramètres σ_{ini} et N_P pour l'AUDyC et des paramètres λ et τ pour le SAKM. Dans la plupart des expériences effectuées, le réglage des paramètres du SAKM s'est révélé plus simple que celui de paramètres de l'AUDyC. Cette étude a permis de faciliter le choix des paramètres des deux algorithmes afin de permettre leur utilisation dans de meilleures conditions (qualité de modélisation, bonne convergence,...).

- **Processus d'apprentissage**

Afin d'évaluer nos algorithmes, leurs procédures d'apprentissage ont été expérimentées dans la partie 5.3 avec des données de simulation. Dans les situations d'apparition de classes (après changement brusque ou dérive rapide), les algorithmes AUDyC et SAKM sont très efficaces pour la création de nouvelles classes. Leurs procédures d'adaptation ont prouvé leurs bonnes capacités d'auto-adaptation dans le contexte du suivi de classes évolutives. Les deux algorithmes prennent en compte les phénomènes de fusion mais seul l'AUDyC est capable de réaliser la scission de classes. Leurs mécanismes d'élimination de classes parasites ont également été testés avec succès. Outre la qualité de modélisation et l'absence d'un mécanisme de scission pour le SAKM, les procédures d'apprentissage des deux algorithmes offrent des résultats similaires pour les classes gaussiennes. En revanche, la modélisation adaptative de classes complexes est mieux effectuée par le SAKM.

- **Robustesse au bruit**

Les performances des deux classifieurs dynamiques ont également été testées en environnement bruité. Ces expériences ont révélé que le SAKM est moins robuste au bruit que l'AUDyC. En effet, trop de bruit dans les données peut affecter la modélisation du SAKM. Ceci est dû à la procédure d'adaptation utilisant la méthode du gradient. Pour atténuer ce phénomène, un réglage minutieux du ratio d'apprentissage est nécessaire. La modélisation de l'AUDyC est moins perturbée par le bruit. Cependant, son temps d'exécution se trouve fortement augmenté.

- **Complexité algorithmique**

Les expériences menées dans la partie 5.4 ont permis d'analyser le temps d'exécution de l'AUDyC et celui du SAKM. Ces expériences ont montré que le temps d'exécution de l'AUDyC croît avec la dimension D de données, le nombre J de prototypes et le nombre L de données. Les complexités trouvées sont satisfaisantes. Pour les expériences effectuées avec le SAKM, le temps de calcul évolue linéairement en fonction des grandeurs D, M, τ et L. Ce qui est cohérent avec la complexité théorique (sans la fusion). Par ailleurs, les calculs de complexité théorique et les expériences menées ont montré que l'algorithme SAKM est plus rapide que l'AUDyC.

- **Modélisation de classes**

Les résultats d'expériences effectuées dans ce chapitre ont confirmé les bonnes capacités de modélisation du SAKM. De même, la modélisation gaussienne avec l'approche multimodale pour l'AUDyC a donné des résultats satisfaisants. Mais, on note que lorsque la distribution des données n'est pas gaussienne, l'AUDyC n'est pas à l'abri de problèmes d'optima locaux. Ces défauts de modélisation se manifestent le plus souvent lors du suivi des classes complexes. Dans ces situations, le SAKM offre des modèles de meilleure qualité. Par ailleurs, pour les grandes dimensions, la qualité de modélisation des deux algorithmes se dégrade à cause du "fléau de la dimensionnalité".

5.6 Exemples d'applications : Systèmes d'Aide à la Décision

Dans cette partie, nous proposons d'utiliser nos algorithmes de classification dynamique pour la mise au point de **Systèmes d'Aide à la Décision** (SAD). Les SAD sont de nos jours très recherchés pour la prise de décision dans beaucoup d'applications (surveillance de processus, reconnaissance vocale, aide au diagnostic médical,...). Cependant, l'implémentation en ligne d'un SAD opérationnel est un travail qui pose des difficultés majeures. Ces difficultés sont pour la plupart liées au comportement évolutif des processus réels. Sur la figure 5.6.1, nous proposons la description générale du SAD pour les applications en ligne. Il est composé de trois modules : le premier a pour rôle l'extraction des informations pertinentes caractérisant l'état du processus, le second permet l'apprentissage de modes (ou classes) et enfin le dernier est le module de décision et d'interprétation des résultats. L'extraction des informations pertinentes (Module 1) et la prise de décision (Module 3) sont facilitées grâce aux connaissances d'un expert du processus. Dans un cadre général, l'état de fonctionnement de processus est décrit par des données non-stationnaires.

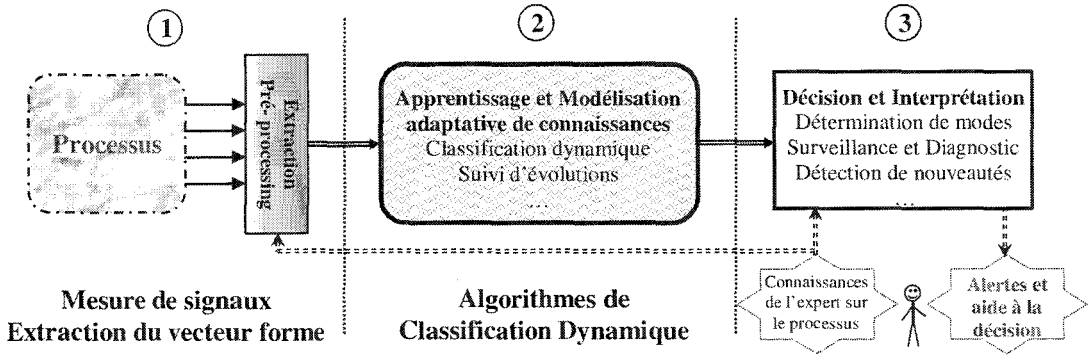

Figure 5.6.1 : Description générale d'une Système d'Aide à la Décision (SAD)

Les algorithmes de classification dynamique que nous avons développés sont utilisés pour la construction du module 2 (figure 5.6.1). Afin de montrer les capacités de nos algorithmes pour la mise au point de systèmes d'aide à la décision dans les problèmes réels, deux validations sont exposées dans cette partie. La section 5.6.1 présente une application de surveillance en ligne d'un processus thermique. Dans la section 5.6.2, nous avons proposé un système de suivi de processus non-stationnaire.

5.6.1 Application à la surveillance d'un processus thermique

Ce travail a été réalisé dans le cadre du projet n°0274032 de l'ADEME (Agence de l'Environnement et de la Maîtrise de l'Énergie). Ce projet co-dirigé par Messieurs Sylvain Lalot et Stéphane Lecoeuche est effectué dans le cadre d'un partenariat entre l'École d'Ingénieurs du Pas-de-Calais (EIPC), l'École de Mines de Douai (EMD), le laboratoire LME de l'Université de Valenciennes, le laboratoire LAGIS de l'Université de Lille 1 et la société Vulcanic. La problématique essentielle de ce projet consiste à détecter l'encrassement des principaux composants (Réchauffeur, Échangeur, Filtre) d'un thermorégulateur (figure 5.6.2).

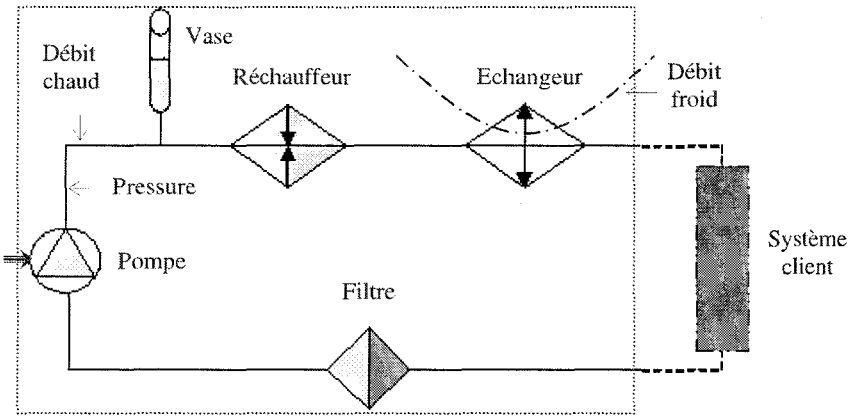

Figure 5.6.2 : Description du processus thermique (thermorégulateur)

Précisons que le prototype final du système de surveillance implémenté sur la maquette est élaboré avec l'algorithme AUDyC. L'interface Homme/Machine est réalisée avec le logiciel Python [Bertier, 2004]. A partir des données issues du thermorégulateur, nous exposons dans cette partie les résultats de l'AUDyC et ceux du SAKM. Grâce à cette application, les deux algorithmes sont validés sur des données réelles.

5.6.1.1 Mise au point du système de surveillance

Le travail effectué a d'abord consisté en la recherche du vecteur caractéristique qui représente à chaque instant, l'état de fonctionnement du processus thermique. Ce vecteur a été établi en utilisant les pertes de pression ΔP au niveau de chaque composant de telle sorte que : $X = \left[\Delta P_d / \Delta P_{Pompe}\right]_d$ avec $d = Rechauffeur, Echangeur, Filter, System\ client$. Ce vecteur de 4 dimensions, est sensible à l'encrassement de chaque composant. L'encrassement est en effet une dégradation lente (obstruction progressive) occasionnant des pertes de pression au sein du composant. A partir des données acquises sur le processus, l'algorithme de classification dynamique effectue la modélisation en ligne des modes de fonctionnement du processus. Le système de surveillance est complété avec des outils de détection couplés au classifieur dynamique (Module 3, figure 5.6.1). Ces outils ont pour objet de détecter l'encrassement qui se traduit par une dérive lente du mode de fonctionnement du processus.

Dans la pratique, le système de surveillance effectue l'apprentissage en ligne des modes du processus au fur et à mesure que les données se présentent. L'apparition d'un nouveau mode est révélée par la création d'une nouvelle classe dans l'espace de décision. Ce mode est labellisé grâce aux connaissances d'un expert du processus, puis mémorisé par le système d'apprentissage. Parallèlement, le module de détection analyse l'évolution du mode courant par rapport aux modes déjà connus. L'une des trois situations suivantes est alors possible :

- Le mode courant reste similaire au mode de fonctionnement normal : pas d'alerte
- Le mode courant évolue vers un mode de défaillance connue : détection de défaut.
- Apparition brusque d'un nouveau mode ou évolution lente vers un mode inconnu : apprentissage.

Grâce à cette méthodologie, le système de surveillance enrichit sa base de connaissances sur le processus tout au long de son utilisation. Et en même temps, il effectue la détection en cas de défaut à partir du suivi et de l'analyse du mode courant. Comme tout système d'apprentissage, l'efficacité de ce SAD augmente avec les expériences accumulées.

Néanmoins, pour illustrer les résultats de la validation de ce système de surveillance sur le thermorégulateur, nous exposons séparément la phase d'apprentissage des quelques modes du processus, puis celle du suivi du mode courant et de la détection de défauts. Pour les besoins de validation, l'encrassement est simulé par la fermeture progressive de vannes placées à proximité de chacun des composants du thermorégulateur.

5.6.1.2 Apprentissage en ligne de modes de fonctionnement

Dans cette section, nous présentons les résultats d'apprentissage de quelques modes du processus. Ce sont les modes les plus intéressants pour la détection d'encrassement. Il s'agit notamment du mode de fonctionnement normal et des modes de défauts correspondant à l'encrassement du réchauffeur, de l'échangeur et du filtre. L'apprentissage du mode normal M_{FN} est effectué pendant le fonctionnement normal du thermorégulateur. Ensuite, en bloquant brusquement la vanne située au niveau du réchauffeur pendant quelques instants, on apprend ainsi le mode d'encrassement réchauffeur M_{R_enc}. En effectuant la même manipulation pour l'échangeur et le filtre, le système apprend leurs modes d'encrassement respectifs M_{E_enc} et M_{F_enc}. Les résultats de l'AUDyC sont présentés sur la figure 5.6.3.a et ceux du SAKM sur la figure 5.6.3.b.

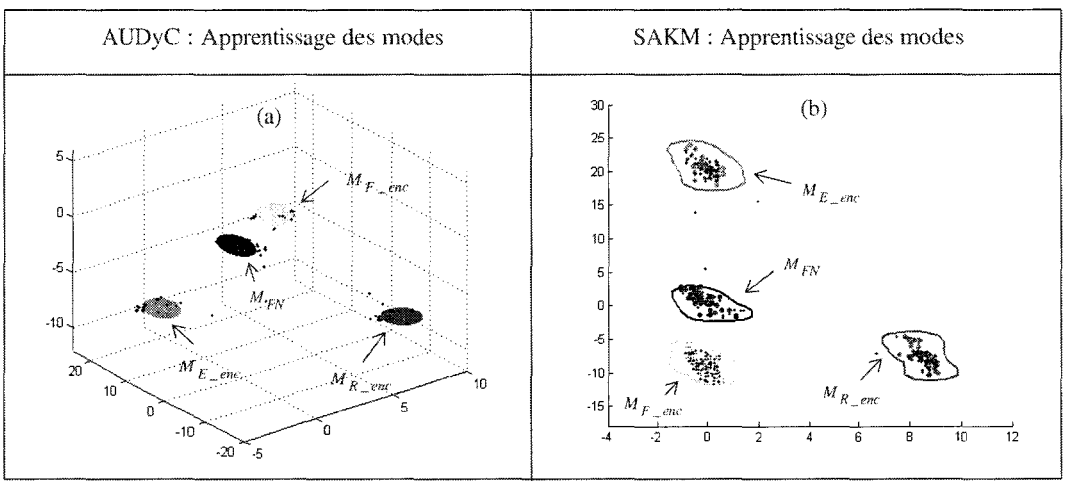

Figure 5.6.3 : Apprentissage des modes de défauts (Encrassement) avec l'AUDyC et le SAKM. /AUDyC : $\sigma_{ini} = 0,4$; $N_p = 200$ /SAKM : $\lambda = 1$; $\eta = 0,15$; $\tau = 50$ / $N_{amb} = 10$; $N_{\min} = 50$; $T = 100$.

5.6.1.3 Suivi de modes et détection d'encrassement

Les données non-stationnaires acquises en ligne sur le processus thermique sont incorporées séquentiellement pour l'adaptation du modèle de classification. De cette façon, le classifieur dynamique effectue la modélisation continue du mode de fonctionnement courant

M_{FC} du processus. Dans les conditions normales, le mode courant reste plus ou moins confondu avec le mode normal M_{FN}. Lorsqu'une dégradation (encrassement) se produit sur l'un des composants du thermorégulateur, le mode courant s'éloigne progressivement du mode normal et évolue vers le mode de défaut caractérisant le composant dégradé. Le niveau d'encrassement des composants du processus est ainsi évalué en utilisant un rapport d'inter-distances entre modes. Celui-ci est établi comme suit :

$$I_{R\_enc} = \frac{\left\|\bar{X}_{FN} - \bar{X}_{R\_enc}\right\|}{\left\|\bar{X}_{FC} - \bar{X}_{R\_enc}\right\|}, \quad I_{E\_enc} = \frac{\left\|\bar{X}_{FN} - \bar{X}_{E\_enc}\right\|}{\left\|\bar{X}_{FC} - \bar{X}_{E\_enc}\right\|}, \quad I_{F\_enc} = \frac{\left\|\bar{X}_{FN} - \bar{X}_{F\_enc}\right\|}{\left\|\bar{X}_{FC} - \bar{X}_{F\_enc}\right\|} \qquad 5.1$$

où I_{R_enc} est l'indicateur de l'encrassement du réchauffeur et $\bar{X}_{FN}$ représente le centre du mode de fonctionnement normal. Les autres grandeurs s'identifient de façon analogue avec leurs indices respectifs. Dans le cas de la modélisation de l'AUDyC, le centre de chaque mode correspond à la moyenne du prototype gaussien (classe mono-prototype), tandis que pour le SAKM, le centre de chaque mode est la moyenne des vecteurs support de la classe.

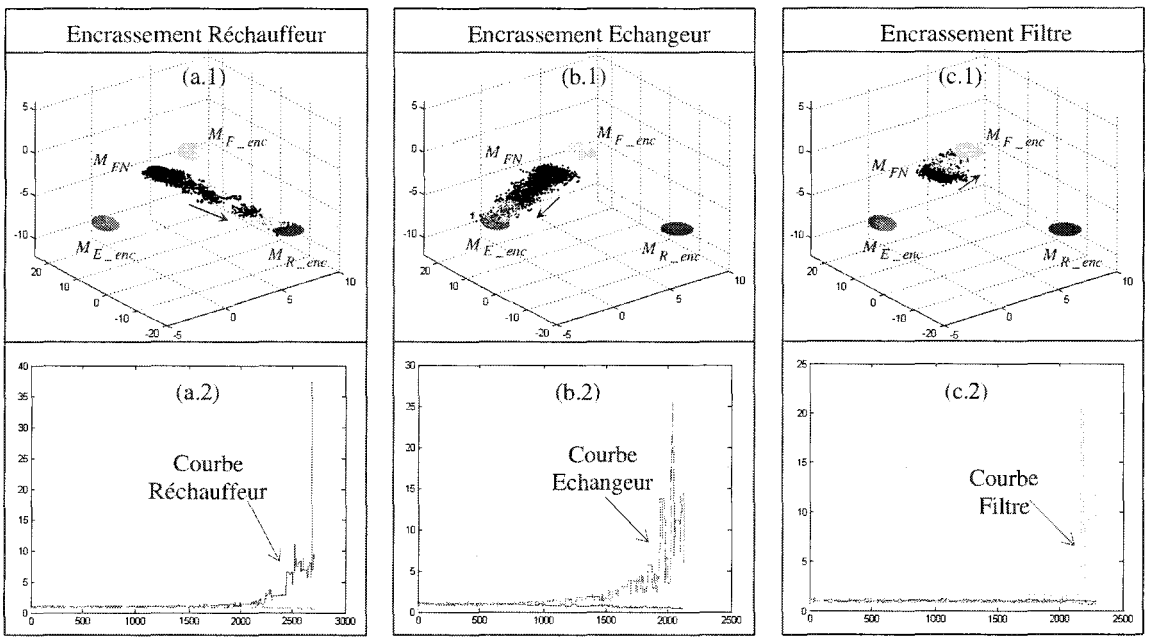

Figure 5.6.4 : Suivi d'évolution du mode de fonctionnement avec l'algorithme AUDyC et mesure de l'encrassement de chaque composant du thermorégulateur.

Les figures 5.6.4.a.1, 5.6.4.b.1 et 5.6.4.c.1 montrent l'évolution du mode courant dont le suivi continu est effectué par l'algorithme AUDyC. Les figures 5.6.4.a.2, 5.6.4.b.2 et 5.6.4.c.2 présentent l'évolution des indicateurs d'encrassement pour chaque composant. Les résultats similaires produits par le SAKM sont présentés sur les figures 5.6.5. A partir des indicateurs d'encrassement, la détection peut se faire en utilisant un seuil ou avec plus de robustesse en utilisant le test de Cusum. Pour plus de précisions sur les outils de détection, le lecteur peut consulter [Amadou, 2002; Amadou & Lecoeuche., 2005b; Lecoeuche & Lalot, 2005].

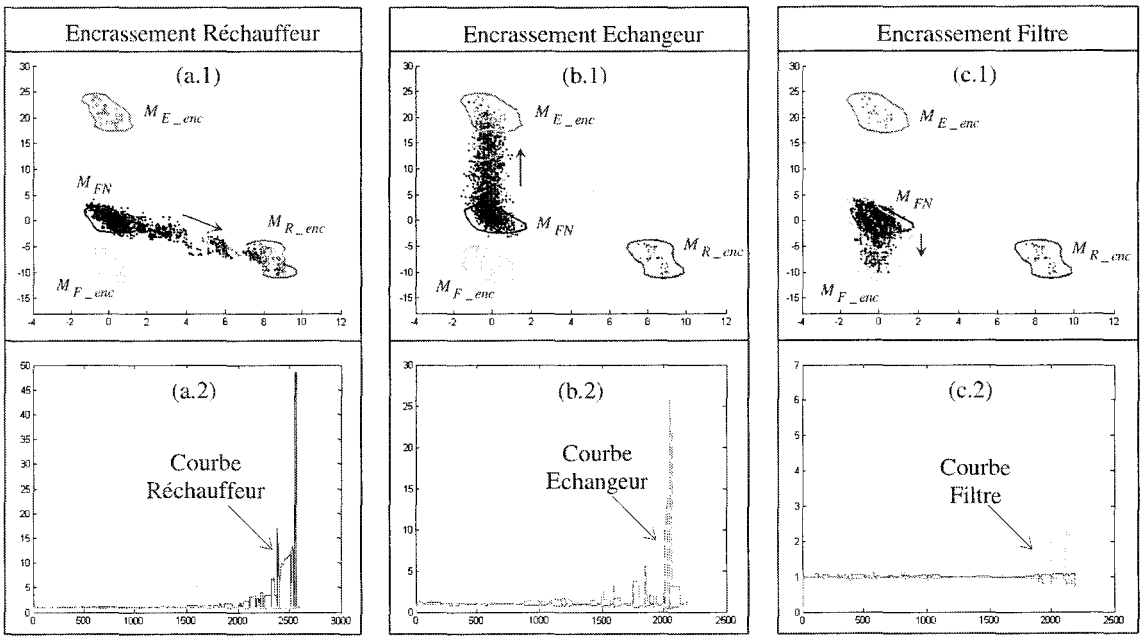

Figure 5.6.5 : Suivi d'évolution du mode de fonctionnement avec l'algorithme SAKM et mesure d'encrassement de chaque composant du thermorégulateur

Les algorithmes développés pour la classification dynamique et les outils de détection ont permis la mise au point d'un SAD opérationnel sur un processus thermique (maquette ADEME). Les résultats des expériences menées pour la détection d'encrassement de trois composants (Réchauffeur, Échangeur, Filtre) du processus thermique sont satisfaisants. Rappelons que l'approche de surveillance adoptée pour les applications en ligne consiste en l'apprentissage continu des modes et parallèlement au suivi et à l'analyse du mode courant. Grâce à cette méthodologie, il est possible d'étendre les capacités de surveillance du système à la détection d'un nombre plus important de défaillances du processus (exemples : rupture de tuyau, usure de pompe, ...). Il faut cependant noter deux limites essentielles à cette approche :

- La première est liée à la modélisation de l'état de fonctionnement du processus. En effet, la mise en œuvre d'un vecteur forme fidèle de l'état du processus nécessite au préalable la connaissance de toutes les informations pertinentes caractérisant l'état du processus. Nul doute que cette tâche est très difficile. Or, la réussite de cette phase d'extraction est une condition primordiale au succès de la surveillance.
- La seconde concerne les pannes et dégradations susceptibles de se produire sur le processus. En réalité, celles-ci ne sont pas dénombrables. D'une part, il est difficile de prévoir le comportement du système lorsque deux ou plusieurs pannes se produisent en même temps. D'autre part, le système de surveillance est incapable de construire une base d'apprentissage exhaustive lui permettant de détecter toutes les défaillances possibles du processus.

Malgré ces limites, notre système de surveillance en ligne est un SAD intéressant qui permet de réduire considérablement les coûts de maintenance en réduisant les délais de détection et en localisant les défauts.

5.6.2 Suivi du mode de fonctionnement d'un processus non-stationnaire

Dans cette section, nous proposons la mise en œuvre d'un système de suivi de processus non-stationnaires. Ce travail est effectué en collaboration avec Monsieur Guillaume Mercère du laboratoire LAII de Poitiers. Le principe retenu consiste à coupler une technique d'identification récursive au classifieur dynamique AUDyC. Un outil d'identification appelée EIVPM (Extended Instrumental Variables Propagator Method) a été développé dans les travaux de thèse de [Mercère 2004]. L'EIVPM effectue une estimation récursive d'un modèle linéaire qui permet d'extraire un vecteur forme modélisant l'état du processus. Ce vecteur est ensuite utilisé par l'algorithme AUDyC qui effectue la modélisation adaptative et le suivi du mode de fonctionnement du processus [Mercère & al., 2006; Lecoeuche & al, 2006].

On présente tout d'abord la simulation du processus non-stationnaire dont l'état est identifié par l'EIVPM. Ensuite, on montre les capacités de ce SAD à suivre les commutations et les évolutions de ce processus.

5.6.2.1 Simulation du processus : génération des données

La technique EIVPM est basée sur l'approche des sous-espaces pour estimer en ligne les matrices d'état d'un modèle linéaire dans un contexte bruité [Mercère, 2004]. Pour la présente expérience, le processus non-stationnaire est simulé par le modèle linéaire :

$$x(t+1) = A_i x(t) + \begin{bmatrix} 0 & 0 & 0.5 \\ 0 & -0.6 & 0 \end{bmatrix}^T \tilde{u}(t) + w(t)$$

$$\tilde{y}(t) = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} x(t); \quad y(t) = \tilde{y}(t) + v(t)$$

où $\tilde{u}$ et $\tilde{y}$ sont respectivement les vecteurs des entrées et des sorties, x le vecteur d'état, w le bruit d'état et v un bruit de mesure en sortie. Les entrées de l'EIVPM sont des bruits blancs gaussiens de variance 1. Le processus est perturbé par un bruit de mesure en sortie coloré de variance 0,03, ainsi que par un bruit d'état coloré de variance 0,05 [Lecoeuche, 2006]. L'évolution du processus simulé est obtenue au travers de la variation des valeurs propres théoriques de la matrice A (figure 5.6.6).

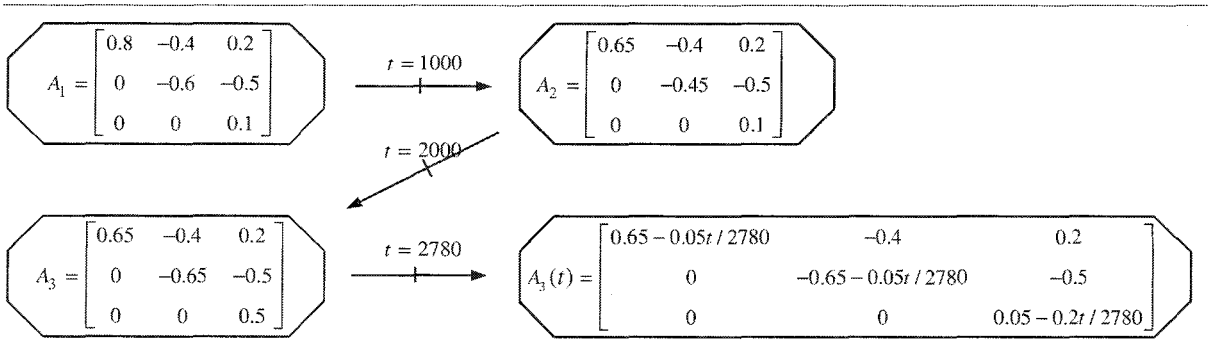

Figure 5.6.6 : Évolution de la matrice A : variation des valeurs propres théoriques

De façon récursive, l'EIVPM identifie le processus et détermine les matrices correspondant à la forme d'état du modèle linéaire. Le vecteur forme caractérisant l'état de fonctionnement du processus est constitué par les valeurs propres vp_k^t de la matrice $\hat{A}$ estimée instantanément par l'EIVPM (figure 5.6.7). Ce vecteur de dimension est ainsi défini : $X_t = [vp_1^t \; vp_2^t \; vp_3^t]^T$.

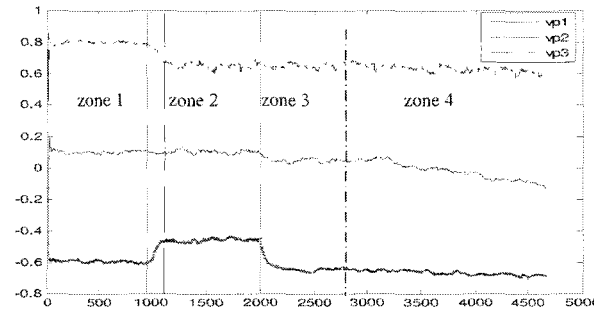

Figure 5.6.7 : Les valeurs propres de la matrice $\hat{A}$ estimée en ligne avec l'algorithme EIVPM

5.6.2.2 Suivi du mode de fonctionnement

Au départ de l'expérience, les deux algorithmes sont initialisés mais aucune connaissance *a priori* excepté l'ordre du processus n'est nécessaire et aucun mode n'est connu. Le vecteur X_t extrait en ligne par l'EIVPM est directement envoyé à l'AUDyC qui se charge du calcul et de la mise à jour du mode de fonctionnement du processus.

- Commutation de modes de fonctionnement

A la première acquisition, l'algorithme AUDyC initialise le premier mode (mode 1) de fonctionnement du processus dans son état de départ (figure 5.6.8.b.1). Ce mode est adapté par l'AUDyC au fur et à mesure que les données arrivent. Il se produit alors une dérive rapide de données jusqu'à stabilisation dans la mode 2. Une nouvelle classe est alors créée (figure 5.6.8.c.1). De la même façon, une dérive rapide de données a lieu du mode 2 vers le mode 3 qui est à son tour créé et adapté (figure 5.6.8.d.1). Notons enfin que d'autres commutations peuvent se produire de façon à revenir aux modes déjà appris. Dans de telles situations, l'AUDyC retrouve les modes déjà appris par le suivi du mode courant.

- Évolution lente d'un mode de fonctionnement

Ce cas montre une évolution lente du mode de fonctionnement du processus. Cette évolution se produit sur le mode 3 (mode courant) (figure 5.6.8.b.2). Au cours de cette évolution, l'AUDyC prend en compte toutes les variations de ce mode en adaptant son modèle au travers de l'intégration des nouvelles données. L'AUDyC effectue ainsi le suivi continu de l'évolution de modes (figures 5.6.8.b.2, 5.6.8.c.2 et 5.6.8.d.2).

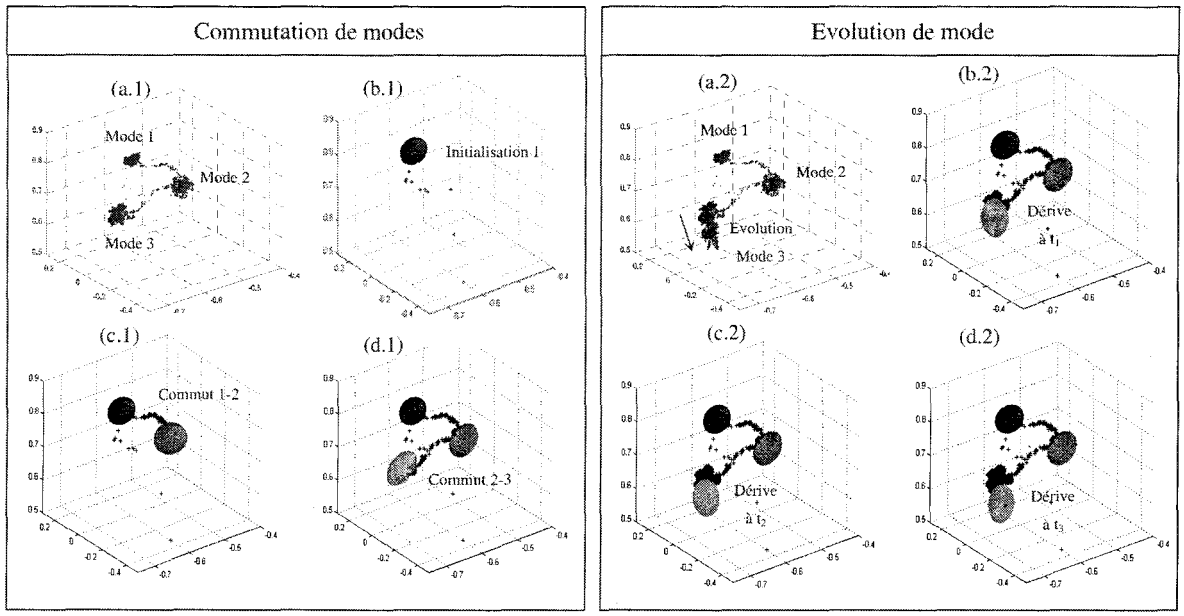

Figure 5.6.8 : Suivi de commutations et d'évolution du mode de fonctionnement d'un processus non-stationnaire en utilisant l'AUDyC. Paramètres : $\sigma_{ini} = 0,05$; $N_P = 500$; $N_{amb} = 5$; $N_{\min} = 30$.

- Détermination en ligne du mode de fonctionnement du processus

A travers ces premiers résultats, la figure 5.6.8 montre que le mode de fonctionnement du processus est déterminé à chaque instant en prenant en compte toutes les évolutions, notamment les commutations de modes caractérisées par les sauts de classes ainsi que les dérives de classes occasionnées par les évolutions de modes au cours du temps.

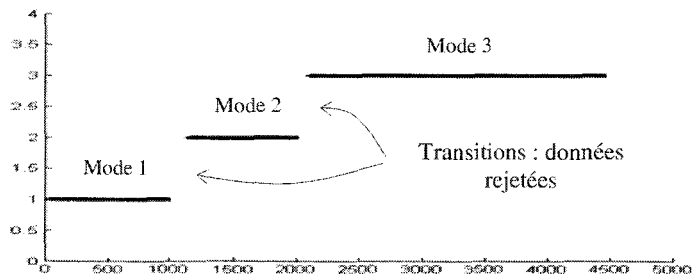

Figure 5.6.9 : Détermination du mode de fonctionnement courant du processus.

Les résultats de cette expérience montrent l'efficacité du SAD pour le suivi de processus non-stationnaires. Validée dans cette expérience sur des données de simulation, cette approche ouvre des perspectives intéressantes pour la modélisation des systèmes hybrides [Lecoeuche & al, 2006].

5.7 Conclusion

Dans le présent chapitre, nous avons validé les classifieurs dynamiques AUDyC et SAKM sur des données de simulation et une application réelle.

Les expériences réalisées ont permis de faciliter l'utilisation de chacun des nos algorithmes à travers la maîtrise du réglage de leurs paramètres. Ensuite, elles ont montré les capacités auto-adaptatives de nos classifieurs dynamiques en environnement de données non-stationnaires. Les tests de complexité effectués ont prouvé que le temps d'exécution en fonction des variables concernées est égal – sinon inférieur – au temps théorique estimé à travers le calcul du nombre des opérations.

Dans la dernière partie de ce chapitre deux Systèmes d'Aide à la Décision (SAD) sont mis au point pour les applications réelles. En utilisant l'AUDyC et le SAKM, un premier SAD est conçu et validé avec succès pour la surveillance d'un processus thermique. Le second SAD ouvre quant à lui des perspectives intéressantes pour le suivi en ligne des processus non-stationnaires.

En se basant sur les résultats expérimentaux et notre expérience de deux algorithmes, nous conseillons l'utilisation de l'AUDyC pour les données distribuées suivant la loi gaussienne et dont la dimension est faible. Pour le traitement de données dont la distribution est inconnue et/ou celles de grande dimension, il est préférable d'utiliser l'algorithme SAKM.

Conclusion et Perspectives

Conclusion

Le travail effectué dans cette thèse a montré les besoins d'algorithmes de classification dynamique capables d'apprendre des connaissances et de suivre leurs évolutions. Bien qu'il existe de nombreux outils de classification performants, peu de travaux sont consacrés à la problématique de données non-stationnaires et de suivi d'évolutions. Or, cette problématique se retrouve dans beaucoup d'applications réelles et pose des difficultés nombreuses et variées. La majorité de ces difficultés est liée au comportement évolutif des processus.

Afin de traiter cette problématique, une étude bibliographique a été consacrée aux techniques avancées utilisées en Traitement de Signal et en Reconnaissance de Formes. Celle-ci a révélé les bonnes performances de modélisation de deux méthodes statistiques. Ce sont les méthodes de modèles de mélange et les méthodes à noyau. Les premières sont dotées d'un formalisme théorique intéressant dans un contexte de classification multi-classe. Les secondes sont très attractives pour leurs bonnes capacités de généralisation et leurs résultats expérimentaux. Cette étude a apporté les outils nécessaires pour l'élaboration de modèles de classification capables de représenter la partition de données. Cependant, celle-ci est susceptible d'évoluer dans le temps. La mise à jour de la partition dynamique se fait à l'aide de règles d'adaptation récursives. Deux groupes de méthodes récursives sont étudiés. Le premier concerne les méthodes du gradient qui sont très utilisées en apprentissage incrémental. Le second regroupe les méthodes de mise à jour exacte que nous avons tentées de formaliser dans ce mémoire.

La première contribution de cette thèse est une description générique proposée pour la conception de classifieurs dynamiques. Le développement de cette description générique est inspiré des rares algorithmes développés pour la classification de données non-stationnaires. Cette description générique est établie sous forme d'un réseau de neurones à architecture évolutive. Afin de lui conférer des capacités d'auto-adaptation, elle est construite en quatre procédures d'apprentissage : création, adaptation, fusion, et évaluation. Deux algorithmes sont développés à partir de cette description générique.

Le premier est une nouvelle version de l'algorithme AUDyC. Cet algorithme est développé avec un formalisme plus élaboré. Il utilise un modèle de mélange inspiré du mélange gaussien multimodal et est décrit suivant l'approche possibiliste. Ce modèle de classification offre deux avantages principaux : L'algorithme est capable d'approximer des classes complexes par la combinaison de plusieurs prototypes gaussiens. L'approche possibiliste confère à l'AUDyC de bonnes aptitudes d'apprentissage en ligne. Plus précisément, cette approche convient aux problèmes d'apprentissage en environnement ouvert où toutes les classes ne sont pas connues *a priori*. L'approche probabiliste n'est pas adaptée pour de tels

problèmes. Par ailleurs, l'AUDyC est doté de nouvelles procédures de fusion et de scission de classes. Grâce à ces nouvelles procédures, les problèmes d'optima locaux (défauts de modélisation) sont moins fréquents. En plus de ces contributions, la convergence et la complexité de cette nouvelle version de l'AUDyC ont été étudiées.

Le second algorithme SAKM, développé suivant la même description générique, est une contribution originale pour les SVM et méthodes à noyau. Il convient aux problèmes de classification séquentielle, multi-classe et adaptative. Grâce aux capacités de généralisation de fonctions d'apprentissage à noyau, le modèle de classification du SAKM s'adapte aux formes complexes. Le processus d'apprentissage de SAKM est conçu avec les procédures introduites dans la description générique. Mais, il ne dispose pas de mécanisme de scission. Enfin, la convergence théorique du SAKM a été démontrée et sa complexité évaluée.

Plusieurs résultats expérimentaux ont montré les capacités d'apprentissage et d'auto-adaptation des deux algorithmes AUDyC et SAKM. Ils effectuent la mise à jour de leurs modèles de classification afin de prendre en compte les évolutions de la partition dynamique. Ces évolutions se manifestent par divers phénomènes : apparition de classes, évolution de classes, fusion de classes, ... etc. Par ailleurs, les complexités pratiques sont analysées. Elles sont bornées par les complexités polynomiales d'ordre $O(D^3 \times J^2 \times L)$ pour l'AUDyC et d'ordre $O(D \times \tau^2 \times M \times L)$ pour le SAKM.

Après cette étude, une comparaison des deux algorithmes a été menée d'un point de vue théorique puis pratique. Celle-ci a montré que l'algorithme SAKM est plus adapté (comparativement à l'AUDyC) pour le traitement de données de distribution inconnue ou celles de dimension relativement grande. Pour les données de faible dimension dont la distribution dans chaque classe peut être approximée par la loi gaussienne, il est préférable d'utiliser l'algorithme AUDyC. En effet, le processus d'apprentissage de cet algorithme est plus élaboré que celui du SAKM. L'AUDyC a l'avantage de disposer d'une procédure de mise à jour exacte et d'une procédure de scission de classes.

Enfin, deux Systèmes d'Aide à la Décision (SAD) ont été conçus en utilisant les classifieurs dynamiques développés dans cette thèse. Le premier SAD est un système de surveillance validé sur un processus thermique. Cette application a également permis de valider nos deux classifieurs AUDyC et le SAKM sur des données réelles. Le second SAD est un système de suivi en ligne de processus non-stationnaires. Dans ce mémoire de thèse, ce système a été validé sur un processus simulé.

Perspectives

Nos travaux de thèse ouvrent plusieurs perspectives de recherche et d'applications. De manière pragmatique, ces travaux apportent une contribution à la classification dynamique. On peut apporter de nombreuses améliorations aux classifieurs dynamiques proposés dans ce mémoire. Pour la poursuite des travaux de recherche, on s'intéresse particulièrement aux perspectives suivantes :

- Développer une *procédure de scission* pour compléter le processus d'apprentissage du SAKM. L'objectif consiste à détecter et à scinder toute fonction d'apprentissage à noyau modélisant deux ou plusieurs groupes de données distinctes en une seule classe. C'est une problématique difficile à résoudre.
- Introduire une *procédure de mise à jour exacte* pour l'algorithme SAKM. L'objectif consiste à améliorer la convergence de l'algorithme. En effet, avec la méthode du gradient stochastique, la procédure d'adaptation du SAKM offre des approximations successives de supports de distribution dans l'espace RKHS.
- Développer des règles d'apprentissage récursives pour la *mise à jour de la forme exponentielle*. Compte tenu que la plupart des lois de densités utilisées dans les modèles de mélange se décline de la forme exponentielle, ce travail permettra d'étendre l'utilisation de l'AUDyC à une plus large variété de lois de densités.
- Trouver des solutions adaptées pour l'AUDyC et le SAKM qui permettront de résoudre, dans une certaine mesure, le problème du "*fléau de la dimensionnalité*". C'est en effet un problème très difficile dont sont victimes la plupart des algorithmes de classification utilisés en grande dimension.

Au terme de ce travail de thèse, une grande diversité des domaines d'application s'ouvre à nos classifieurs dynamiques à travers la mise au point de Systèmes d'Aide à la Décision (SAD). On cite principalement les perspectives d'application suivantes :

- *Supervision de processus industriels* : La mise au point de systèmes de surveillance et de diagnostic en ligne permettant d'une part, la détection de défauts et d'autre, la localisation de défaillances qui se produisent sur un processus industriel.
- *Suivi de processus non-stationnaires* : A partir de la modélisation et du suivi en ligne d'un processus hybride, il est possible de déterminer à chaque instant le mode de fonctionnement du processus.
- *Détection de ruptures* : En exploitant leurs capacités de détection de changement et de création de nouvelles classes, les classifieurs dynamiques peuvent servir à la mise au point de systèmes de détection de ruptures (dans les signaux audio par exemple).

Références bibliographiques

Amadou-Boubacar, H. (2002). Outils pour la Détection de Changement de Comportement d'un Système. DEA Automatique & Informatique Industrielle, Université des Sciences et Technologies de Lille (USTL). France.

Amadou-Boubacar, H., Lecoeuche, S. (2005a). A new kernel-based algorithm for online clustering. Springer-Verlag, Duch, W. and al. (Eds): International Conference on Neural Networks (ICANN05), pp: 583-588. Warsaw, Poland.

Amadou-Boubacar, H., Lecoeuche, S. (2005b). System Drifts Monitoring using Neural Supervision System. IASTED proceedings of Artificial Intelligence and Applications (AIA05, ed.), pp: 803-808. Innsbruck, Austria.

Amadou-Boubacar, H., Lecoeuche, S., Maouche, S. (2005a). Self-Adaptive Kernel Machine: Online Clustering in RKHS. IEEE proceedings, IJCNN05.

Amadou-Boubacar, H., Lecoeuche, S., Maouche, S. (2005b). AUDyC Neural Network using a new Gaussian Densities Merge Mechanism. In the 7th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA05, ed.), pp: 155-158. Coimbra, Portugal.

Amari, S. (1993). Backpropagation and stochastic gradient descent method. Neurocomputing, vol. 5(4-5), pp. 185–196, 1993.

Arbib, M A. (1987). Levels of modeling of mechanisms of visually guided behavior. Behavioral and Brain Sciences, vol. 10, pp: 407-465.

Balakrishnan, S., V. (2003). Fast Incremental Adaptation Using Maximum Likelihood Regression and Stochastic Gradient Descent. 8th European Conference on Speech Communication and Technology, Geneva, Switzerland.

Ball G. and Hall D. (1965). ISODATA: a novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park, CA.

Belleman, R. E. (1961). Adaptive Control Processes. Princeton University Press.

Ben-Hur, A., Hava, A., Siegelmann, T., Vapnik, V. (2002). Support Vector Clustering. Journal of Machine Learning Research, vol. 2, pp: 125-137.

Bennett, J., Campbel, C., (2000). Support Vector Machines: Hype or Hallelujah. SIGKDD Explorations, vol. 2(3), pp: 1-13.

Bertier, S. (2004). Surveillance et diagnostic d'un procédé thermique basé sur une modélisation de classes dynamiques. DEA Automatique & Informatique Industrielle, Université des Sciences et Technologies de Lille (USTL). France.

Bezdek, J. C. (1982). Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York.

Bezdek, J. C. (1984). FCM: The fuzzy c-means clustering algorithm. Computers and Geosciences, vol. 10, pp: 191-203.

Bilmes, J. A. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden Markov models. Technical Report, University of Berkeley, USA.

Bishop, C. M. (1995). Neural Networks for Pattern Recognition. Clarendon Press, Oxford, UK.

Borer, S. (2003). New Support Vector Algorithms for Multi-categorical Data: Applied to Real-Time Object Recognition. Ph. D. Thesis, EPFL, Lausanne Swiss.

Bottou, L. (1998). Online Algorithms and Stochastic Approximations. In Online Learning and Neural Networks, David Saad, ed., Cambridge, UK.

Bottou, L. (2004). Stochastic Learning. In Advanced Lectures on Machine Learning, no. LNAI 3176 (Olivier Bousquet, O., Luxburg, U. V. eds.), pp: 146-168, Berlin.

Bottou, L., Bengio, Y. (1995). Convergence properties of the k-means algorithms. MIT press, Advances in Neural Information Processing Systems, vol. 7, pp: 585-592.

Boudaoud, A., N. (1998). Conception d'un système de diagnostic adaptatif en ligne pour la surveillance des systèmes évolutifs. Thèse de l'Université de Compiègne, France.

Bousquet, O., Boucheron, S., Lugosi, G. (2004). Introduction to Statistical Learning Theory. Advanced Lectures on Machine Learning Lecture Notes in Artificial Intelligence 3176, pp: 169-207. (Eds.) Bousquet, O., Luxburg, U. V., Rätsch, G. . Springer, Heidelberg, Germany.

Bruges, C. (1998). A tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knwoledge Discovery, vol. 2, pp: 121-167.

Cauwenberghs, G., Poggio, T. (2001). Incremental and Decremental Support Vector Machine Learning. MIT Press, Cambridge, MA, Advances in Neural Information Systems, pp: 409-415.

Celeux, G., Christien, S., Forbes, F., Mkhadri, A. (1999). A component-Wise EM algorithm for mixtures. Technical Report INRIA: Institut National de Recherche en Informatique et en Automatique.

Celeux, G., Govaert, G. (1995). Gaussian parsimonious clustering models. Pattern Recognition, vol. 5, pp: 781-793.

Chaudhuri, D., Chaudhuri, B. B., Murthy, C. A. (1992). A new split-and-merge clustering technique. Pattern Recognition Letters, vol. 13(6), pp: 399-409.

Chelcea, G., Bertrand, P., Trousse, B. (2004). Un Nouvel Algorithme de Classification Ascendante 2-3 Hiérarchique. Reconnaissance des Formes et Intelligence Artificielle (RFIA'04), vol. 3, pp: 1471-1480, Toulouse, France.

Chen, P. H., Jin, C. J., Schölkopf, B. (2005). A tutorial on nu-support vector machines. Appl. Stochastic Models Bus. Ind., vol. 21, pp: 111–136.

Cornuéjols, A. (1993). Training Issues in Incremental Learning. Proceedings of the AAAI Spring Symposium, Stanford University, March 23-25.

Cornuéjols, A., Miclet, L. (2002). Apprentissage artificiel - concepts et algorithmes. Edition Eyrolles.

Coulon, R. (2002). Apprentissage par Renforcement utilisant des Réseaux de Neurones, avec des applications au contrôle moteur. Thèse de Doctorat, Institut National Polytechnique de Grénoble (INPG), France.

Cover, T. M., Hart, P. E. (1967). Nearest neighbour pattern classification. IEEE Transactions on Information Theory, vol. 13, pp: 21-27.

Cristianini, N., Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines. Cambridge University Press, Cambridge, UK.

Cuevas, A., Febrero, M., Fraiman, R. (2001). Cluster analysis: a further approach based on density estimation. Computational Statistics & Data Analysis, vol. 36, pp: 441-459.

Dang, M. V. (1998). Classification de Données Spatiales: Modèles Probabilistes et Critères de Partitionnement. Thèse de Doctorat, Université de Technologie de Compiègne.

Dempster, A. P., Laird, N. M., Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM Algorithm (with) discussion. Journal of the Royal Statistical Society series B, vol. 39, pp: 1-38.

Deng, D., Kasabov, N. (2003). On-line pattern analysis by evolving self organizing maps. Neurocomputing, vol. 51, pp: 87-103.

Denœux, T. (2000). A Neural Network Classifier Based on Dempster-Shafer Theory. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 30(2), pp: 1083-4427.

Desobry, F. (2004). Méthodes à noyau pour la détection de ruptures. Thèse de Doctorat, Université de Nantes, École Centrale de Nantes, France.

Desobry, F., Davy, M., Doncarli, C. (2005). An online kernel change detection algorithm. IEEE Transactions on Signal Processing, vol. 53(8-2), pp: 2961-2974.

Dessalles, J. L. (1996). L'ordinateur génétique. Edition Hermès, Paris.

Devroye, L., Wise, G. L. (1980). Detection of abnormal behavior via nonparametric estimation of the support. SIAM Journal on Applied Mathematics, vol. 38(3) pp: 480-488.

Dongxin, X. (1999). Entropie and Information Potential for Neural Computation. PhD Thesis, University of Florida, USA.

Doye, D., Sontakke, T. (2002). Speech Recognition Using Modified General Fuzzy Min-Max Neural. The Pennsylvania State University CiteSeer Archives, USA.

Dubuisson, B. (1990). Diagnostic et Reconnaissance des Formes. Edition Hermès, Paris.

Dumitrescu, D. (2000). Genetic Algorithms and Evolution Strategies, Blue Publishing House, Cluj-Napoca.

Einmahl, J., Mason, D. (1992). Generalized quantile processes. The Annals of Statistics, vol. 20(2), pp: 1062-1078.

Eltoft, T. (1998). A new Neural Network for Cluster-Detection-and-Labeling. IEEE Trans. on Neural Networks, vol. 9(5), pp: 1021-1035.

Evgeniou, T., Pontil, M., Poggio, T. (2000). "Regularization Networks and Support Vector Machines." Advances in Computational Mathematics, vol. 13(1), pp: 1–50.

Fabiani, P. (1994). A New Approach in Temporal Representation of Belief for Autonomous Observation and Surveillance Systems. European Conference on Artificial Intelligence – ECAI, pp: 391-395.

Fabiani, P. (1996). Représentation dynamique de l'incertain et stratégie de prise d'information pour un système autonome en environnement évolutif. Thèse de doctorat, Ecole Nationale Supérieure de l'Aéronautique et de L'Espace – ENSAE.

Fiesler, E. (1994). Neural Networks Formalization and Classification. Computer Standard & Interfaces, Special Issue on Neural Networks Standards, John Fulcher (Ed.), vol. 16(3).

Fleury, S. (1998). Représentation et classification évolutives pour le traitement automatique du langage naturel. 1998. Ecole Nationale Supérieure de Fontenay St- Cloud.

Friedman, J. (1996). Another approach to polychotomous classification. Technical Report, Stanford University, USA.

Fukunaga, K. (1990). Introduction to Statistical Pattern Recognition, 2nd edition. Academic Press, New York.

Gabrys, B., Bargiela, A. (2000). General Fuzzy Min-Max Neural Network for Clustering and Classification. IEEE Trans. Neural Networks, vol. 11(3), pp: 769-783.

Genevieve B. O., Todd, K. L. (1993). Weight space probability densities in stochastic learning: II. transients and basin hopping times. In Hason, S., Cowan, J. and Giles, L., editors, Advances in Neural Information Processing Systems 5. Morgan Kaufmann, San Mateo, CA. vol. 61.

Gentile, C. (2001). A new approximation maximal margin classification algorithm. Journal of Machine Learning Research, vol.2, pp: 313-242.

Golub, G., Loan, V. (1996). Matrix computations. 3rd édition, The J ; Hopkins University Press, London.

Gordon, P. (1965). Théorie des chaines de Markov finies et ses applications. Edition Dunod, Paris.

Gorunescu, R., Dumitrescu, D. (2003). Evolutionary Clustering using an Incremental Technique. Studia Univ. babes, Bolyai, Informatica, vol. XL 8(2).

Govaert, G. (2003). L'analyse des données. Edition Hermès, Paris.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, Biometrika, vol. 82, pp: 711–732.

Gretton, A., Desobry, F. (2003). Online one-class nu-svm, an application to signal segmentation. IEEE Proceedings, ICASSP'03, vol. 2, pp: 709-712.

Guermeur, Y. (2002). Combining Discriminant Models with New Multi-Class SVMs. Pattern Analysis and Applications, vol. 5(2), pp: 168-179.

Guermeur, Y., Paugam-Moisy, H. (1999). Théorie de l'apprentissage de Vapnik et SVM, Support Vector Machines. In Sebban, M. and Venturini, G. editors, Apprentissage automatique, pp: 109-138, Edition Hermès, Paris.

Halgamuge, S. K., Wang, L. (2005). Classification and Clustering for Knowledge Discovery. Springer Verlag, Series: Studies in Computational Intelligence, vol. 4.

Hartigan, J. A. (1975). Clustering Algorithms. Wiley, New York.

Hartley, HO (1958). Maximum likelihood estimation from incomplete data. Biometrics, vol. 14, pp: 174-194

Hebb, D. O. (1949). The Organization of Behavior. Wiley, New York.

Herbrich, R. (2002). Learning Kernel Classifiers. MIT Press, Cambridge, MA.

Hogg, R. V., Whittinghill, D. C. (2001). A Little Uniform Density With Big Instruction Potential. Jounal of Statistics Education, vol. 9(2).

Holland, J. (1975). Adaptation in Natural and Artificial Systems. The University of Michigan Press.

Jain, A., Duin, R., Mao, J. (2000). Statistical pattern recognition: a review. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22(1), pp: 4-37.

Jin, B., Hurson, A. R., Miller, L. (1991). Neural Network-Based Decision Support for Incomplete Database Systems. Knowledge Acquisition and Performance Analysis. Conference on Analysis of Neural Network Applications, pp:62-75.

Jodouin, J. F. (1994). Les Réseaux Neuromimétiques: Modèles et applications. Editions Hermès, Paris.

Kasabov, N. K. (2003). Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines. Springer Verlag edition.

Kaufman, L. (1999). Solving the Quadratic Programming Problem Arising in Support Vector Classification. Advances in Kernel Methods - Support Vector Learning, edited by Scholkopf, B., Burges, C. J. C., Smola, A. J., MIT Press, Cambridge, MA, pp: 147-167.

Kijima, M. (1997). Markov Processes for Stochastic Modeling. Edition Chapman & Hall.

Kivinen, J., Smola, A. J., Williamson, R. C. (2004). Online Learning with Kernels. IEEE Trans. on Signal Processing, vol. 100(10).

Kohonen, T. (2001). Self-organizing maps. Springer-Verlag, Berlin.

Kuhn, H. W., Tucker, A. W. (1950). Nonlinear programming. In J. Neyman, editor, Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, pp: 481-492.

Lapidot, I., Guterman, H., Cohen, A. (2002). Unsupervised Speaker Recognition Based on Competition Between Self-Organizing Maps. IEEE Trans. Neural Networks, vol. 13(4), pp: 877- 887.

Lecoeuche S., (2006). Apprentissage Récursif: Quelques méthodes d'identification et de classification pour le suivi de systèmes évolutifs. Rapport de HDR, Université de Sciences et Technologies de Lille, France.

Lecoeuche S., Lalot, S. (2005). Neural network based online detection of fouling in a Water Circulating Temperature Controller (WCTC). Proceeding of 6th Int. Conf. on Heart Exchanger Fouling and Cleaning. IRSEE, Germany.

Lecoeuche S., Lurette, C. (2003). Auto-adaptive and Dynamical clustering Neural Network. Springer, ICANN'03 proceedings, pp: 350-358.

Lecoeuche, S., Lurette, C. (2004). New supervision architecture based on on line modelization of non stationary data. Neural Computing and Applications, vol. 13(4), pp: 323-338.

Lecoeuche, S., Mercère, G., Amadou-Boubacar, H. (2006). Modelling of Non-stationary Systems based on a Dynamical Decision Space. 14th IFAC Symposium on System Identification, (SYSID06), Newcastle, Australia.

Leski, J. (2004). Fuzzy c-varieties/elliptotypes clustering in reproducing kernel Hilbert space. Fuzzy Sets and Systems, vol. 141(2), pp: 259-280.

Letac, G., Mora, M. (1990). Natural Real Exponential Family with Cubic Variance Functions. Ann. Statist, vol. 18, pp: 1-37.

Liu, Q., Levinson, S., Wu, Y., Thomas, S., Huang, S. (2000). Interactive and Incremental Learning via a Mixture of Supervised and Unsupervised Learning Strategies. In Proc. of Joint Conf. on Information Systems (JCIS'00), NJ, USA.

Ljung, L. (1999). System identification. Theory for the user. Practice Hall Information, 2nd édition.

Loonis, P., Locteau, H. (2004). Planification markovienne du choix d'agents de classification pour la poursuite de classes évolutives. Reconnaissance des Formes et Intelligence Artificielle (RFIA'04), vol. 2, pp:1039-1048, Toulouse, France.

Lurette, C. (2003). Développement d'une technique neuronale auto-adaptative pour la classification dynamique des données évolutives: Application à la supervision d'une presse hydraulique. Thèse de doctorat Université des Sciences et Technologies de Lille, France.

McLachlan, G. J. and Peel, D. (2000). Finite Mixture Models. Wiley, New York.

McQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In proceedings of the Fifty Berkeley Symposium on Mathematical Statistics and Probabilty, vol. 1, pp: 281-297.

Mendel J. M., and Fu, K. S. (1970). Adaptive, Learning and Pattern Recognition Systems: Theory and Applications. Academic Press, New York.

Meneganti, M., Saviello, M. F. S., Tagliaferri, R. (1998). Fuzzy neural networks for classification and detection of anomalies. IEEE Trans. Neural Networks, vol. 9.

Merwe, R., Wan, E., A. (2001). Efficient Derivative-Free Kalman Filters for Online Learning. ESANN'01 proceedings - European Symposium on Artificial Neural Networks Bruges, Belgium, ISBN 2-930307-01-3, pp: 205-210.

Mercère, G. (2004). Contribution à l'identification récursive des systèmes par l'approche des sous-espaces. Thèse de doctorat, Université des Sciences et Technologies de Lille, France.

Mercere, G., Amadou-Boubacar, H., Lecoeuche, S., Bako, L. (2006). Suivi de Systèmes Non-stationnaires basé sur une approche de Reconnaissance des Formes. Conférence Internationale Francophone d'Automatique (CIFA06, ed.). Bordeaux, France. Accepté.

Micchelli, C. A. (1986). Interpolation of scattered data: Distance matrices and conditionally positive definite functions. Constructive Approximation, vol. 2, pp: 11–22.

Milgram, J., Sabourin, R., Cheriet, M. (2004) Système de classification à deux niveaux de décision combinant approche par modélisation et machines à vecteurs de support. Colloque International Francophone (CIFED'04). La Rochelle, France, pp: 25-29.

Minoux, M. (1983). Programmation Mathématique : Théorie et Algorithmes. Dunod édition, Tome 1, Paris.

Mobley R. K. (1992). La maintenance prédictive, Organisation Industrielle. Edition Masson.

Moon, C., Kim, J., Choi, G., Seo, Y. (2002). An efficient genetic algorithm for the traveling salesman problem with precedence constraints. European Journal of Operational Research, vol. 140(3), pp: 606-617.

Mora, M. (1986). Famille Exponentielle et Fonctions Variance. Thèse de doctorat, Université Paul-Sabatier, Toulouse.

Mouchaweh, M. S., Devillez, A., Lecolier, G. V., Billaudel, P. (2002). Incremental learning in Fuzzy Pattern Matching. Fuzzy Sets and Systems vol. 132(1): 49-62.

Mouchaweh, M. S., (2002). Conception d'un système de diagnostic adaptatif et prédictif basé sur la méthode Fuzzy Pattern Matching pour la surveillance en ligne des systèmes évolutifs. Thèse, Laboratoire d'Automatique et de Micro-électronique (LAM), Université de Reims-Champagne-Ardenne, France.

Muller, D. W. (1992). The excess mass approach in statistics. Technical Report Beitrage zur Statistik (Nr. 3), Universitaat Heidelberg.

Murray, K. S. (1995). Learning as Knowledge Integration. Ph.D thesis, University of Texas, Austin, TX.

Musicant, D. R. (2000). Data Mining via Mathematical Programming and Machine Learning. Ph. D. Thesis, Computer Sciences Department, University of Wisconsin, Madison.

Neal, R., Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variantes. in M. I. Jordan (eds) Learning in Graphical Models Dordrecht: Kluwer Academic, pp: 355-368.

Nobile, A., Fearnside, A. (2005). Bayesian finite mixtures with an unknown number of components: the allocation sampler. University of Glasgow, U. K. Technical Report.

Nougier, J. P. (1993). Méthodes de calcul numérique. 3e Edition, Masson, Paris.

Osorio F. S. (1998). INSS : Un système hybride neuro-symbolique pour l'apprentissage automatique constructif. Thèse de l'Institut National Polytechnique de Grenoble I.N.P.G., Laboratoire LEIBNIZ – IMAG

Pilla, R. S., Lindsay, B. G. (2001). Alternative EM methods for nonparametric finite mixture models. Biometrika, vol. 88, pp: 535-550.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. Advances in kernel methods: support vector learning, MIT Press, Cambridge, MA.

Polikar, R., Udpa, L., Udpa, S., and Honavar, V. (2001). Learn++: An Incremental Learning Algorithm for Multi-Layer Perceptron Networks. IEEE Transactions on Systems, Man, and Cybernetics, vol. 31(4) pp. 497-508.

Portera, F. (2005). Loss Functions and Structured Domains for Support Vector Machines. Technical Report UBLCS-2005-08, Departement of Computer Science, University of Bologna.

Pronzato, L., Pazman, A., (2004).Moindres carrés pondérés récursifs dans les modèles de régression à variance paramétrée. Journées Francaises de Statistiques, Monpellier, France.

Richardson, S., Green, P. J. (1997). On Bayesian analysis of mixtures with an unknown number of components (with discussion), J. R. Stat. Soc. Ser. B, vol. 59, pp: 731–792.

Robbins, H., Monro, S. (1951). A Stochastic Approximation Model. Annals of Mathematical Statistics, vol. 22, pp: 400-407.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, vol. 65, pp: 386-408.

Saint-Jean, C. (2001). Classification paramétrique robuste partiellement supervisée en reconnaissance des formes. Thèse de l'université de la Rochelle UFR L3I,France.

Salomon, J. (2001). Support Vector Machines for Phoneme Classification. Master of science, School of Artificial Intelligence, University of Edinburgh.

Samé, A. B. (2005). Modèles de mélange et classification de données acoustiques en temps réel. Thèse de l'Université de Technologie de Compiègne (UTC), Compiègne, France.

Samé, A. B., Ambroise, C., Govaert, G. (2004). A Mixture Model Approach for On-line Clustering. Physica-Verlag Springer, COMPSTAT'04 symposium.

Santos, O. F. (1998). INSS: un système hybride neuro-symbolique pour l'apprentissage automatique constructif. Thèse de doctorat, Institut National Polytechnique de Grenoble – INPG, France.

Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A. (2001). Estimating the Support of a High-Dimensional Distribution. Neural Computation, vol. 13, pp: 1443-1471.

Schölkopf, B., Smola, A. (2002). Learning with Kernels. MA: MIT Press, Cambridge.

Schölkopf, B., Smola, A., Williamson, R., Bartlett, P., (2000). New Support Vector algorithms. Neural Computation, vol. 12, pp: 1207-1245.

Schölkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., Vapnik, V. (1997). Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Trans. Sign. Processing, vol. 45, pp: 2758 – 2765.

Shaohua, K., Shellappa, R. (2004). Kullback-Leibler Distance between Two Gaussian Densities in Reproducing Kernel Hilbert Space. Chicago DC, ISIT.

Silva, V. R., Khatib, W., Fleming, P. J. (2005). Performance optimization of gas turbine engine. Engineering Applications of Artificial Intelligence, vol. 18(5), pp: 575-583.

Simpson, P. K. (1992). Fuzzy min-max neural networks. Part 1: classification. IEEE Transactions on Neural Networks, vol. 3, pp: 776-786.

Simpson, P. K. (1993). Fuzzy min-max neural networks. Part 2: clustering. IEEE Transactions on Fuzzy Systems, vol. 1(1), pp: 32-45.

Smola, J. A., Schölkopf, B. (2004). A tutorial on support vector regression. Statistics and Computing, vol. 14, pp: 199-222.

Stephens, M. (2000a). Dealing with label-switching in mixtures models. Journal of Royal Statistical Society Series B.

Stephens, M. (2000b). Bayesian analysis of mixture models with an unknown number of components. an alternative to reversible jump methods. Ann. Statist. 28(1), pp: 40–74

Sun, B. Y., Huang, D. S. (2003). Support vector clustering for multiclass classification problems. Evolutionary Computation, vol. 2, pp: 1480-1485.

Tax, D., Duin, R. (1999). Support vector domain description. Pattern Recognition. Letters, vol. 20, pp: 1991–1999.

Thiria, S., Lechevallier, Y., Gascuel, O., Canu, S. (1996). Statistiques et Méthodes neuronales, Statistique et Mét., Edition Dunod.

Tikhonov, A. N., Arsenin, V. Y. (1977). Solution of ill-posed problems. W. H. Winston, Washingtown, DC.

Tong, S. (2001). Active learning: Theory & Applications. PhD Thesis, Department of Computer Science of Stanford University, USA.

Tu, Y.K. (2003). Hierarchical Text Classification using One-Class-SVM. Thesis of Computer Science & Information Ingineering, National Cheng Kung University, Taïwan, China.

Ueda, N., Nakano, R., Gharhamani, Z., Hinton, G. (2000). SMEM algorithm for mixture models. Neural Computation, vol. 12, pp: 2109-2128.

Vannoorenberghe, P., Smets, Ph. (2005). Partially supervised learning by a credal EM approach. ECSQARU'05, Barcelone, Spain.

Vapnik, V. (1995). The Nature of Statistical Learning Theory. Springer-Verlag, New York.

Vapnik, V. (1998). Statistical Learning Theory. John Wiley & Sons Inc, New York.

Vapnik, V. (1999). An Overview of Statistical Learning Theory. IEEE Trans. Neural Networks, vol. 10(5), pp: 988-999.

Vasseur, C., Lakel, R., Hublin, P. (1988). Estimation récursive des caractéristiques d'une classe en reconnaissance de formes. Université de Lille 1, Centre d'Automatique, Villeneuve d'Ascq, France.

Vijayakumar, S. (1998). Computational Theory of Incremental and Active Learning for Optimal Generalization, Doctoral Thesis, Tokyo Institute of Technology, 95D38108.

Voegtlin, T. (2002). Recursive self-organizing maps. Neural Networks, vol. 15(8-9), pp: 979-991.

Walther, G. (1997). Granulometric smoothing. The Annals of Statistics, vol. 25(6), pp: 2273-2299.

Weston, J., Watkins, C. (1999). Multiclass support vector machines. In M. Verleysen, editor, Proceedings of ESANN99. D. Facto Press.

Yager, R. R., Fedrizzi, M., Kacprzyk, J. (1994). Advances in the Dempster–Shafer Theory of Evidence. Wiley, New York.

Zadeh, L. A. (1965). Fuzzy sets. Informations and Control, vol. 8, pp: 338-353.

Zhang, B., Zhang, C., Yi, X. (2004). Competitive EM Algorithm for Finite Mixture Models. Pattern Recognition, vol. 37(1), pp: 131-144.

Zhang, Z., Chen, C., Sun, J., Chan, K. L. (2003). EM algorithms for Gaussian mixtures with split-and-merge operation. Pattern Recognition, vol. 36, pp: 1973-1983.

ANNEXES

Annexe A.1 : Mise à jour récursive de la forme exponentielle

Le mélange gaussien et bien d'autres lois de densités (exponentielle, poisson, Dirichlet, gamma,...) utilisées dans les modèles de mélange font partie d'une même famille connue sous le nom de forme exponentielle [Mora, 1986]. On peut montrer qu'en appliquant directement la méthode du gradient stochastique sur le critère de vraisemblance de la forme exponentielle, on obtient les formules de mise à jour récursive des paramètres de divers modèles de densité. Mais avant, commençons par définir la forme exponentielle.

Définition A.1 [Samé, 2004] **:** Une famille de densités de probabilité dans le cas des variables continues ou des lois de probabilité dans le cas de variables discrètes $\{P(X,\Theta);\ X \in \chi, \Theta \in \Omega\}$ est une forme exponentielle à n ($n \in \mathbb{N}_*$) paramètres si $P(X,\Theta)$ peut s'écrire de la façon suivante :

$$P(X;\Theta) = \exp\left(\Lambda(\Theta)^T \mathrm{T}(X) - a(\Lambda(\Theta)) + b(X)\right) \qquad \text{A.1}$$

Λ : fonction à valeurs dans $\Re^n$, appelée statistique exhaustive naturelle.

T : fonction à valeurs dans $\Re^n$, appelée paramètre naturel.

a et b : fonctions à valeurs dans $\Re$.

A partir de l'expression de la forme exponentielle, on écrit simplement le critère de log-vraisemblance de la manière suivante :

$$L[\Theta] = Log\left(P(X;\Theta)\right) = \Lambda(\Theta)^T \mathrm{T}(X) - a(\Lambda(\Theta)) + b(X) \qquad \text{A.2}$$

En utilisant le gradient stochastique généralisé, la maximisation du critère (A.2) passe par le calcul du gradient de $L[\Theta]$ pour trouver le terme d'adaptation $H(\Theta, X)$ de telle sorte que :

$$\tilde{E}_\chi[H(\Theta,X)] = \frac{\partial L[\Theta]}{\partial \Theta} = \frac{\partial \Lambda}{\partial \Theta}\left(\mathrm{T}(X) - \frac{\partial a}{\partial \Lambda}\right) \qquad \text{A.3}$$

La formule itérative de mise à jour du modèle s'écrit donc :

$$\Theta^{t+1} = \Theta^t + \eta \frac{\partial \Lambda^t}{\partial \Theta}\left(\mathrm{T}(X_t) - \frac{\partial a^t}{\partial \Lambda}\right) \qquad \text{A.4}$$

Pour un modèle de densité choisi, il suffit d'identifier les fonctions Λ, T et a de la forme exponentielle pour déterminer la formule itérative de mise à jour des paramètre de ce modèle. Dans le cas du modèle de mélange gaussien par exemple, [Samé 2004] montre que :

$$T(X)=\left[\sum_{i=1}^{L} X_i,\ \sum_{i=1}^{L} X_i X_i^T\right] \qquad \text{A.5}$$

$$\Lambda\left(\bar{X},\Sigma\right)=\left[\Sigma^{-1}\bar{X},\ \frac{1}{2}diag\left(\Sigma^{-1}\right)-\Sigma^{-1}\right] \qquad \text{A.6}$$

$$a(\Lambda)=\frac{L}{2}\left(\bar{X}\Sigma^{-1}\bar{X}+\log\left(2\pi\right)^{D}\left|\Sigma\right|\right) \qquad \text{A.7}$$

$$b\left(X\right)=0 \qquad \text{A.8}$$

Après dérivation, la formule (A.4) permet d'établir les règles de mise à jour du modèle gaussien à partir de la méthode du gradient stochastique. Ces règles de mise à jour sont utilisées pour la mise en œuvre du CEM incrémental (chapitre 3, équations 3.13, 3.14, 3.15 et 3.16).

Annexe A.2 : Fonction noyau et Espace de Hilbert

A.2.1 Définitions

Le concept d'espace de Hilbert à noyau reproduisant (RKHS) exposé dans le chapitre 2, partie 2.3 repose sur le formalisme des fonctions noyaux. Cet espace est défini à partir d'une structure de représentation vectorielle dotée du produit scalaire et appelée espace de Hilbert.

Définition A.2 : Soit l'ensemble de données $\{X_i\}_{i=1,\ldots,L}$ de l'espace χ, on appelle *fonction noyau* $\kappa(\cdot,\cdot)$ une mesure de similarité *symétrique* définie paire par paire sur l'ensemble de couples de $\chi\times\chi$ vers $\Re$, i.e. $\kappa(X_i,X_k)=\kappa(X_k,X_i)\in\Re$, $\forall(X_i,X_k)\in\chi\times\chi$

Définition A.3 : Une fonction noyau (symétrique) à valeurs réelles est appelée *noyau défini positif* si et seulement si, la condition suivante est vérifiée :

$$(X_i,X_k)\in\chi^2,\ (a,b)\in\Re^2\quad \sum_{i,k}K_{i,k}\geq 0 \qquad \text{A.9}$$

$K_{i,k}=\kappa(X_i,X_k)$, élément d'une matrice carrée de taille L appelée *matrice de Gram.*

En général, l'appellation noyau[1] se réfère tout simplement à un noyau défini positif. Dans la littérature, on retrouve aussi les termes de noyau reproduisant, noyau de Mercer, noyau admissible.

La mesure de similarité exprimée par le produit scalaire est un noyau définie positif i.e. $\kappa(X_i,X_k)=\kappa(X_k,X_i)=\langle X_1,X_2\rangle$. Cet opérateur est une mesure de similarité adéquate pour l'apprentissage de modèles linéaires. Cependant, la complexité réside dans le traitement de non-linéarités dans la plupart des problèmes. Afin de conférer aux méthodes à noyau des capacités d'apprentissage des modèles non-linéaires, le produit scalaire est défini dans un espace de Hilbert Γ. La projection des données de l'espace χ vers Γ est rendue possible grâce à une transformation non-linéaire ϕ : $\forall X\in\chi,\ \phi(X)=\kappa(\cdot,X)$. Un espace de Hilbert doit au préalable avoir une structure d'espace pré-Hilbertien

Définition A.4 : Un espace *pré-Hilbertien* Γ est un espace vectoriel doté du produit scalaire $\langle\cdot,\cdot\rangle_\Gamma$[2].

Définition A.5 : Un espace de *Hilbert* Γ est un espace pré-Hilbertien complété de la norme $\|\cdot\|_\Gamma$ induite par le produit scalaire tel que pour tout f dans Γ, la norme $\|f\|_\Gamma^2=\langle f,f\rangle_\Gamma$.

---

[1] Nous conservant également cette appellation (noyau) pour qualifier un noyaux défini positif.

[2] $\langle\cdot,\cdot\rangle_\Gamma$ est le produit scalaire défini dans l'espace Γ

Un espace de Hilbert de fonctions $f:\chi \to \Re$ est un espace RKHS s'il existe un noyau $\kappa(\cdot,\cdot)$ telle que la propriété du noyau reproduisant est vérifiée (chapitre 2, définition 2.4).

A.2.2 Initiation au SVM

Une méthode simple de classification peut être décrite en utilisant le noyau produit scalaire. Considérons un ensemble de données $\{X_1,...,X_L\}$ partagées par deux classes distinctes C_1 et C_2 de centres respectifs c_1 et c_2. Supposons $y=1$ les labels des données appartenant à C_1 et $y=-1$ ceux des données affectées à C_2. La représentation géométrique de la figure A.1 interprète la configuration de cette classification. En considérant une donnée X à classer et $c=\frac{1}{2}(c_1+c_2)$ le point à mi-distance entre les centres c_1 et c_2, un classifieur simple peut être construit avec comme mesure de similarité le produit scalaire. La règle de décision de ce classifieur s'exprime simplement par la formule (figure A.1.a) :

$$y=\mathrm{sgn}(<(X-c),(c_1-c_2)>) \qquad \text{A.10}$$

En remplaçant les centres (ou moyennes) c_1 et c_2 par leurs expressions, c'est à dire les moyennes : $c_1=\bar{E}(\{X_i \in C_1\})$ et $c_2=\bar{E}(\{X_i \in C_2\})$, on peut montrer que :

$$y=\mathrm{sgn}\left(\frac{1}{n_1}\sum_{X_i\in C_1}\kappa(X,X_i)-\frac{1}{n_2}\sum_{X_i\in C_2}\kappa(X,X_i)+b_0\right) \qquad \text{A.11}$$

Avec le noyau $\kappa(X,X_i)=<X,X_i>$; n_j la cardinalité de la classe C_j et le paramètre b_0 s'écrit :

$$b_0=\frac{1}{2}\left(\frac{1}{n_1^2}\sum_{X_i,X_j\in C_1}\kappa(X_i,X_j)-\frac{1}{n_2^2}\sum_{X_i,X_j\in C_2}\kappa(X_i,X_j)\right) \qquad \text{A.12}$$

On remarque que cette méthode correspond à un classifieur bayésien séparant les deux classes en utilisant leur densités de probabilité estimée chacune sur une fenêtre de *Parzen* (A.11).

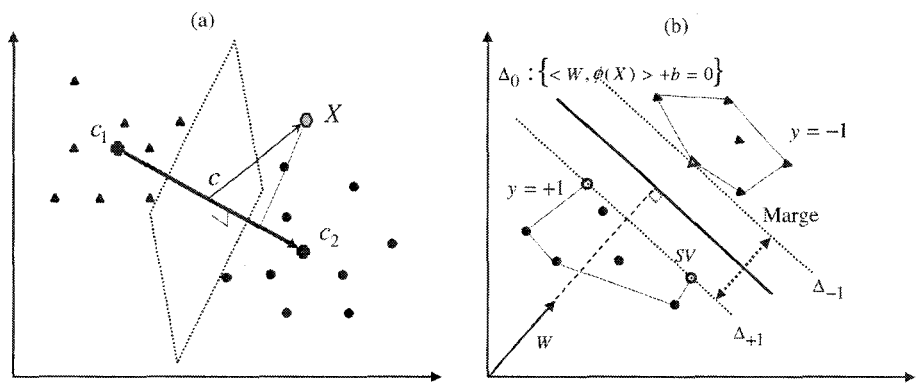

Figure A.1 : (a) Méthode simple de classification binaire. (b) Classification SVM dans RKHS.

Généralisation : La forme du classifieur précédent (A.11) peut se généraliser facilement sous une forme plus flexible et mieux adapté à une large variété des problèmes : c'est le classifieur SVM dont la fonction de décision s'écrit :

$$y = \mathrm{sgn}\left(f(X)\right) = \mathrm{sgn}\left(\sum_{i=1}^{L} \alpha_i \kappa(X, X_i) + b\right) \qquad \text{A.13}$$

Dans l'espace de Hilbert, cette fonction s'exprime par : $f(X) = < W, \phi(X_i) >_\Gamma + b$, $W \in \Gamma$, $b \in \Re$ (figure A.1.b). La fonction de classification optimale doit être celle maximisant la marge séparant les données de deux classes (figure A.1.b). Cette marge entre les hyperplans Δ_{-1} et Δ_{+1} est égale à $1/\|W\|$. Le problème du classifieur SVM correspond donc au problème d'optimisation :

$$\begin{array}{ll} \text{Minimiser} & \frac{1}{2}\|W\|^2 \\ \text{Sous contraintes} & y_i\left(< W, \phi(X) >_\Gamma + b\right) \geq 0 \end{array} \qquad \text{A.14}$$

Annexe A.3 : Formules de mise à jour du modèle gaussien

On considère l'ensemble $\{X_1,...X_L\}$ des données appartenant à un prototype gaussien. On suppose que ces données sont distribuées suivant une loi normale P. En utilisant l'estimateur théorique de densité gaussienne $\hat{E}(\bullet|P)$, le paramètre $\Theta=(\bar{X},\Sigma)$ est estimé par :

$$\Theta=\hat{E}(X_1,...,X_L|P) \rightarrow \begin{cases} \bar{X}=\frac{1}{L}\sum_{i=1}^{L}X_i \\ \Sigma=\frac{1}{L-1}\sum_{i=1}^{L}(X_i-\bar{X})(X_i-\bar{X})^T \end{cases} \quad \text{A.15}$$

Dans cette annexe, on effectue la détermination de la récurrence sur le modèle afin d'établir les formules récursives de mise à jour du prototype gaussien à travers l'ajout et le retrait d'informations.

Ajout et retrait d'information

Considérons $\Theta^N=\hat{E}(\{X_{t-N+1},....,X_t\}|P)$ et $\Theta^{N*}=\hat{E}(\{X_{t-N+2},...,X_{t+1}\}|P)$ les paramètres du prototype gaussien estimés sur une fenêtre glissante de taille N. L'objectif consiste à déterminer la récurrence sur la moyenne et la matrice de covariance au travers de l'ajout et du retrait d'information. Il s'agit donc de réécrire l'estimateur $\hat{E}(\bullet|P)$ sous sa forme récursive :

$$\Theta^{N*}=r'.\Theta^N+\beta'.AR(\Theta^N,X_{t-N+1},X_{t+1}) \quad \text{A.16}$$

où AR : opérateur d'ajout et de retrait d'information sur une fenêtre glissante et r' et β' sont des coefficients scalaires ou matriciels.

- Détermination de la récurrence sur la moyenne

$$\bar{X}^{N*}=\frac{1}{N}\sum_{i=t-N+2}^{t+1} \Rightarrow \bar{X}^{N*}=\frac{1}{N}\left(\sum_{i=t-N+1}^{t}X_i+X_{t+1}-X_{t-N+1}\right)$$

$$\Rightarrow \bar{X}^{N*}=\frac{1}{N}\left(N.\bar{X}^N+X_{t+1}-X_{t-N+1}\right)$$

$$\bar{X}^{N*}=\bar{X}^N+\frac{1}{N}(X_{t+1}-X_{t-N+1}) \quad \text{A.17}$$

Ou encore : $\bar{X}^{N*}=\bar{X}^N+\frac{1}{N}\left(\delta X^+-\delta X^-\right)$ en posant $\delta X^+=X^{t+1}-\bar{X}^N$ et $\delta X^-=X^{t-N+1}-\bar{X}^N$.

On établit l'opérateur $AR(\Theta^N,X_{t-N+1},X_{t+1})=X_{t+1}-X_{t-N+1}$ et les paramètres $r'=1$ et $\beta'=\frac{1}{N}$.

- Détermination de la récurrence sur la matrice de covariance

$$\Sigma^{N*} = \frac{1}{N}\sum_{i=t-N+2}^{t+1}\left(X_i - \bar{X}^{N*}\right)\left(X_i - \bar{X}^{N*}\right)^T$$

$$\Sigma^{N*} = \frac{1}{N}\left[\underbrace{\sum_{i=t-N+1}^{t+1}\left(X_i - \bar{X}^{N*}\right)\left(X_i - \bar{X}^{N*}\right)^T}_{A} + \underbrace{\left(X_{t+1} - \bar{X}^{N*}\right)\left(X_{t+1} - \bar{X}^{N*}\right)^T}_{B} - \underbrace{\left(X_{t-N+1} - \bar{X}^{N*}\right)\left(X_{t-N+1} - \bar{X}^{N*}\right)^T}_{C}\right] \qquad \text{A.18}$$

- Calcul du terme A

En utilisant l'expression de la moyenne (A.17), on a

$$A = \sum_{i=t-N+1}^{t+1}\left(X_i - \bar{X}^N - \frac{1}{N+1}\left(X_{t+1} - X_{t-N+1}\right)\right)\left(X_i - \bar{X}^N - \frac{1}{N+1}\left(X_{t+1} - X_{t-N+1}\right)\right)^T$$

$$A = \sum_{i=t-N+1}^{t+1}\left(X_i - \bar{X}^N\right)\left(X_i - \bar{X}^N\right)^T - \frac{2}{N+1}\sum_{i=t-N+1}^{t+1}\left(X_i - \bar{X}^N\right)\left(X_{t+1} - X_{t-N+1}\right)^T + \frac{N}{(N+1)^2}\left(X_{t+1} - X_{t-N+1}\right)\left(X_{t+1} - X_{t-N+1}\right)^T$$

En remarquant que : $\sum_{i=t-N+1}^{t+1}\left(X_i - \bar{X}^N\right) = 0$ puis en développement, on obtient :

$$A = \sum_{i=t-N+1}^{t+1}\left(X_i - \bar{X}^N\right)\left(X_i - \bar{X}^N\right)^T + \frac{1}{N}\left(\delta X^+ - \delta X^-\right)\left(\delta X^+ - \delta X^-\right)^T$$

$$A = (N-1).\Sigma^N + \frac{1}{N}\left(\delta X^+.\delta X^{+T} - \delta X^+.\delta X^{-T} - \delta X^-.\delta X^{+T} + \delta X^-.\delta X^{-T}\right) \qquad \text{A.19}$$

- Calcul du terme B

$$B = \left(X_{t+1} - \bar{X}^N - \frac{1}{N}\left(\delta X^+ - \delta X^-\right)\right)\left(X_{t+1} - \bar{X}^N - \frac{1}{N}\left(\delta X^+ - \delta X^-\right)\right)^T$$

$$B = \left(\delta X^+ - \frac{1}{N}\left(\delta X^+ - \delta X^-\right)\right)\left(\delta X^+ - \frac{1}{N}\left(\delta X^+ - \delta X^-\right)\right)^T$$

$$B = \frac{1}{N^2}\left((N-1)\delta X^+ + \delta X^-\right)\left((N-1)\delta X^+ + \delta X^-\right)^T$$

$$B = \frac{1}{N^2}\left((N-1)^2\delta X^+.\delta X^{+T} + (N-1)\delta X^+.\delta X^{-T} + (N-1)\delta X^-.\delta X^{+T} + \delta X^-.\delta X^{-T}\right) \qquad \text{A.20}$$

- Calcul du terme C

$$C = \left(X_{t-N+1} - \bar{X}^N - \frac{1}{N}\left(\delta X^+ - \delta X^-\right)\right)\left(X_{t-N+1} - \bar{X}^N - \frac{1}{N}\left(\delta X^+ - \delta X^-\right)\right)^T$$

$$C = \left(\delta X^- - \frac{1}{N}\left(\delta X^+ - \delta X^-\right)\right)\left(\delta X^- - \frac{1}{N}\left(\delta X^+ - \delta X^-\right)\right)^T$$

$$C = \frac{1}{N^2}\left((N+1)\delta X^- + \delta X^+\right)\left((N+1)\delta X^- + \delta X^+\right)^T$$

$$C = \frac{1}{N^2}\left((N+1)^2 \delta X^-.\delta X^{-T} - (N+1)\delta X^-.\delta X^{+T} - (N+1)\delta X^+.\delta X^{-T} + \delta X^+.\delta X^{+T}\right) \qquad \text{A.21}$$

Enfin, à partir des résultats précédents, on a :

$$\begin{aligned}\Sigma^{N*} = \Sigma^N &+ \frac{1}{N-1}\left(\frac{1}{N} + \frac{(N-1)^2}{N^2} - \frac{1}{N^2}\right).\delta X^+.\delta X^{+T} \\ &+ \frac{1}{N-1}\left(\frac{1}{N} + \frac{1}{N^2} - \frac{(N-1)^2}{N^2}\right).\delta X^-.\delta X^{-T} \\ &+ \frac{1}{N-1}\left(-\frac{1}{N} + \frac{N-1}{N^2} + \frac{N+1}{N^2}\right).\delta X^+.\delta X^{-T} \\ &+ \frac{1}{N-1}\left(-\frac{1}{N} + \frac{N-1}{N^2} + \frac{N+1}{N^2}\right).\delta X^-.\delta X^{+T}\end{aligned}$$

Ainsi,

$$\begin{aligned}\Sigma^{N*} = \Sigma^N &+ \frac{1}{N}.\delta X^+.\delta X^{+T} \\ &- \frac{N+1}{N(N-1)}.\delta X^-.\delta X^{-T} \\ &- \frac{1}{N(N-1)}.\delta X^+.\delta X^{-T} \\ &- \frac{1}{N(N-1)}.\delta X^-.\delta X^{+T}\end{aligned}$$

Des calculs précédents, on déduit les formules récursives de mise à jour exacte de la matrice de covariance :

$$\Sigma^{N*} = \Sigma^N + \Delta X B^{-1} \Delta X^T \qquad \text{A.22}$$

avec : $B = \frac{1}{N}\begin{pmatrix} 1 & \frac{1}{N-1} \\ \frac{1}{N-1} & -\frac{N+1}{N-1} \end{pmatrix}$, $\Delta X = \begin{bmatrix} \delta X^+ & \delta X^- \end{bmatrix}$

On en déduit que : $\beta'.AR\left(\Theta^N, X_{t-N+1}, X_{t+1}\right) = \Delta X B^{-1} \Delta X^T$ et $r' = 1$

Annexe A.4 : Réglage des Paramètres AUDyC

Dans cette annexe, certains paramètres de l'AUDyC sont étudiés afin de faciliter leur réglage. Le bruit dans les données n'est pas pris en compte. Le réglage des paramètres est étudié uniquement en prenant en compte la distribution des données. Soit χ_{ech} un échantillon de données, nous allons tout d'abord introduire les hypothèses suivantes :

- Les données $X_i \in \chi_{ech}$ sont générées suivant la loi normale, i.e. densité gaussienne. On suppose que toute les données de χ_{ech} appartiennent à un prototype représentatif P.
- La notion de prototype élémentaire P_e est introduite dans cette étude pour définir le plus petit prototype représentatif, de telle sorte que : $\text{card}(P) \geq \text{card}(P_e)$. En ne tenant compte que de la distribution de données, le prototype élémentaire est celui dont la cardinalité est suffisante pour estimer correctement son modèle, i.e., $\text{card}(P_e) = N_{\min}$.

A partir de la distribution des données, on étudie les conditions d'initialisation de la matrice de covariance, des seuils d'appartenance et des seuils de cardinalité et d'ambiguïté.

A.4.1 La matrice de covariance initiale Σ_{ini}

La matrice de covariance d'un prototype gaussien caractérise la dispersion des données autour de la moyenne de ce prototype. Pour un prototype P, cette matrice est en réalité définie par les écarts entre les données suivant toutes les directions de telle sorte que :

$$\Sigma = \frac{1}{L-1}\sum_{i=1}^{L}(X_i - \bar{X})(X_i - \bar{X})^T = \frac{1}{L-1}\sum_{i=1}^{L}\begin{bmatrix} e_{1i}^2 & \cdots & e_{1i}.e_{Di} \\ \vdots & e_{di}^2 & \vdots \\ e_{1i}.e_{Di} & \cdots & e_{Di}^2 \end{bmatrix} \qquad \text{A.23}$$

Avec : $e_{di} = (x_{di} - \bar{x}_d)$ où $X_i = [x_{1i} \ldots x_{di} \ldots x_{Di}]^T$, $\bar{X} = [\bar{x}_1, \ldots, \bar{x}_d, \ldots, \bar{x}_D]^T$

Pour calculer correctement la matrice de covariance, il est nécessaire que le nombre de données du prototype soit suffisant. Par hypothèse, la notion de prototype élémentaire P_e est introduite pour désigner le plus petit prototype ayant un nombre $N_{\min}$ suffisant de points pour déterminer correctement le modèle. La détermination de la matrice de covariance Σ de prototypes est effectuée de la manière suivante :

$$\begin{aligned} &\text{Si } \text{card}(P) \leq N_{\min} \quad \text{alors } \Sigma \leftarrow \Sigma_{ini} \\ &\text{Si } \text{card}(P) > N_{\min} \quad \text{alors } \Sigma \leftarrow R_{updat}(\Sigma, X_{new}) \end{aligned} \qquad \text{A.24}$$

Σ_{ini} est la matrice de covariance initiale de prototypes. Cette matrice est utilisée pour initialiser les prototypes ayant une cardinalité inférieure à $\text{card}(P_e) = N_{\min}$. La difficulté consiste à trouver une méthode d'estimation de Σ_{ini} à partir d'un échantillon χ_{ech} de données. Sans aucune connaissance *a priori* sur la distribution de données, on suppose que la variance est

identique suivant toutes les directions de l'espace χ. De cette façon, la matrice de covariance initiale Σ_{ini} peut donc s'écrire de la façon suivante :

$$\Sigma_{ini} = \sigma_{ini}^2 \times I_D \quad \text{A.25}$$

[3]

σ_{ini}^2 est la variance (i.e. écart-type au carrée) calculée sur chaque direction pour l'ensemble de données de χ_{ech}. Elle s'exprime comme la moyenne dans chaque direction d :

$$\sigma_{ini}^2 = \sigma_{ini,d}^2 = \tilde{E}_i\left[\left(x_i^d - \overline{x}^d\right)^2\right] = \frac{1}{L}\sum_{i=1}^{L}(x_i^d - \overline{x}^d)^2$$

où x_i^d la $d^{ième}$ coordonnée du vecteur X_i et $\overline{x}^d$ la coordonnée moyenne suivant la direction d. En considérant l_{dist} la moyenne des inter-distances entre les données X_i et la moyenne $\overline{X}$ de l'échantillon, on peut écrire :

$$l_{dist}^2 = \tilde{E}_i\left[\left\|X_i - \overline{X}\right\|^2\right] = \tilde{E}_i\left[\sum_{d=1}^{D}\left(x_i^d - \overline{x}^d\right)^2\right] = \sum_{d=1}^{D}\left[\sigma_{ini,d}^2\right]$$

Or, sachant que l'écart-type est le même dans toutes les directions $\sigma_{ini} = \sigma_{ini,d}$, $\forall d \in [D]$, on :

$$\sigma_{ini}^2 = l_{dist}^2 / D \quad \text{A.26}$$

La détermination de la matrice de covariance initiale Σ_{ini} passe donc par l'estimation de l_{dist}. Pour déterminer l_{dist} nous proposons plusieurs méthodes :

- **Méthode 1**

L'estimation de l_{dist} est effectuée avec l'inter-distance minimale entre les données :

$$l_{dist}^2 = \min_{i \neq j}\left(\left\|X_i - X_j\right\|^2\right), \quad \forall X_i, X_j \in \chi_{ech} \quad \text{A.27}$$

Cette méthode d'estimation a l'inconvénient majeur de donner une valeur trop faible de σ_{ini}^2. Par conséquent, elle augmente le risque de création d'un nombre trop important de prototypes.

- **Méthode 2**

On choisi ici d'estimer l_{dist} avec l'inter-distance maximale entre les données.

$$l_{dist}^2 = \max_{i \neq j}\left(\left\|X_i - X_j\right\|^2\right), \quad \forall X_i, X_j \in \chi_{ech} \quad \text{A.28}$$

Avec cette démarche, il y a un risque important que le volume initial de prototypes nouvellement créés soit trop grand. Ceci causera un problème de sur-apprentissage.

[3] I_D : Matrice identité de dimension D

- **Méthode 3**

L'estimation de l_{dist} est effectué avec la moyenne des inter-distances.

$$l_{dist}^2 = \text{moy}\left(\left\|X_i - X_j\right\|^2\right), \quad \forall X_i, X_j \in \chi_{ech} \qquad \text{A.29}$$

Cette méthode donne des résultats assez satisfaisants. Cependant, elle ne peut marcher que si l'hypothèse de départ selon laquelle toutes les données de l'échantillon χ_{ech} appartiennent au même prototype est satisfaite. Or, cette hypothèse est une hypothèse forte qui est rarement acceptable dans la pratique d'autant plus qu'on ne dispose d'aucune connaissance *a priori* sur l'appartenance de données aux prototypes.

- **Méthode 4**

La présente méthode se base sur la précédente mais recherche au préalable les données de χ_{ech} pouvant appartenir au même prototype pour effectuer l'estimation de l_{dist}. Elle vise donc à éviter l'inconvénient constater dans les méthodes précédentes en éliminant les données trop isolées (figure A.2). Pour cela, on propose d'établir une distance seuil d_{avg} pour détecter et éliminer les données isolées de telle sorte que :

$$d_{avg}^2 = \frac{1}{card(\chi_{ech})}\sum_i d_{\min}^2(X_i) \qquad \text{avec} \qquad d_{\min}^2(X_i) = \min_{i \neq j}\left(\left\|X_i - X_j\right\|^2\right) \qquad \text{A.30}$$

$\chi_{iso} = \left\{X_i \in \chi_{ech} / d_{\min}(X_i) > 3 \times d_{avg}\right\}$, ensemble des données isolées.

$$l_{dist}^2 = \frac{3}{\text{card}(\chi_{ech}) - \text{card}(\chi_{iso})} \sum_{\substack{X_i \in \chi_{ech} \\ X_i \notin \chi_{iso}}} d_{\min}^2(X_i) \qquad \text{A.31}$$

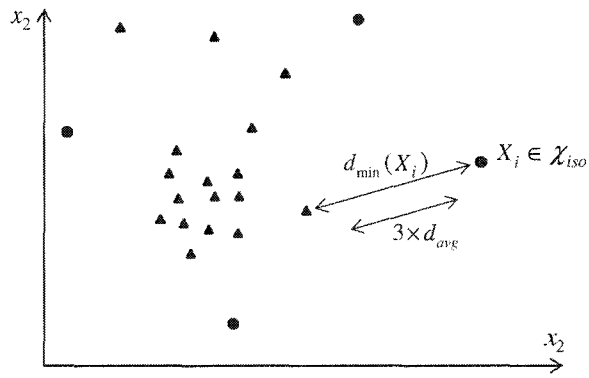

Figure A.2 : Échantillon χ_{ech} de données pour le calcul de σ_{ini}^2. Élimination des données isolées.

Les méthodes d'initialisation 1, 2, et 3 supposent au préalable que toutes les données de l'échantillon χ_{ech} appartiennent au même prototype. Or, cette supposition est une hypothèse forte qui est rarement satisfaite dans la pratique. Et, même avec la satisfaction de cette hypothèse, seule la méthode 3 semble donner de bons résultats. Parmi toutes les méthodes

proposées nous choisissons d'adopter la méthode 4. Elle est insensible au bruit dans les données de l'échantillon et donne des résultats satisfaisants dans la pratique. Il faut cependant noter que la méthode d'initialisation proposée dans cette section ne permet pas de trouver systématiquement la valeur idéale de la matrice de covariance initiale Σ_{ini}. Elle donne une valeur approchée à partir de laquelle, on peut facilement régler Σ_{ini} au travers de quelques essais et corrections. Par exemple, si la valeur idéale de Σ_{ini} est de 1. La méthode d'initialisation peut trouver 1,6. De cette façon, on recherche la valeur de Σ_{ini} au alentours de 1,6 au lieu de 50 (par exemple). Le temps de réglage des paramètres de l'AUDyC est ainsi considérablement réduit.

A.4.2 Les seuils d'appartenance $\mu_{\min}$ et $\mu_{\max}$

L'objectif de l'étude menée dans cette section, consiste à montrer que les seuils d'appartenance $\mu_{\min}$ et $\mu_{\max}$ ne dépendent pas de la distribution des données et peuvent donc être fixé indépendamment de celle-ci.

On considère un prototype P_j de paramètre $\theta(\bar{X}_j, \Sigma_j)$, la fonction d'appartenance relative à ce prototype est définie telle que :

$$\mu_j(X) = \exp\left(-\frac{1}{2}(X-\bar{X}_j)^T(\Sigma_j)^{-1}(X-\bar{X}_j)\right) \qquad \text{A.32}$$

Les seuils $\mu_{\min}$ et $\mu_{\max}$ étant identiques pour tous les prototypes (hyper-sphériques ou hyper-ellipsoïdaux), on peut donc faire leur détermination en supposant que le prototype P_j est hyper-sphérique. Avec cette simplification, on peut écrire :

$$\Sigma_j = \sigma_j^2 \times I_D \qquad \text{A.33}$$

Toutes les données appartenant au prototype P_j se trouvent ainsi dans un contour (hyper-sphérique) tel que : $X \in P_j \Leftrightarrow \|X-\bar{X}_j\| \leq k.\sigma_j$, k est une constante $\Re$. En choisissant un point X_a situé à l'intersection entre le contour de P_j et le premier axe de l'espace χ, on a :

$$X_a - \bar{X}_j = [k.\sigma_j, 0, ..., 0]^T \qquad \text{A.34}$$

A partir des expressions (A.33) et (A.34), l'écriture (A.32) ne dépendra que de k et de p_j de telle sorte qu'on puisse écrire $\mu_j(k) = \mu_j(X)$ de la manière suivante :

$$\mu_j(k) = \exp\left(-\frac{1}{2}\left[k.\sigma_j, 0, ..., 0\right]\left[\frac{1}{\sigma_j^2}.I_D\right]\left[k.\sigma_j, 0, ..., 0\right]^T\right) \quad \text{A.35}$$

$$\mu_j(k) = \exp\left(-\frac{1}{2}k\right) \Rightarrow \begin{cases} \mu_{\min} = \mu_j(k_1) = \exp\left(-\frac{1}{2}k_1\right) \\ \mu_{\max}(k_2) = \exp\left(-\frac{1}{2}k_2\right) \end{cases} \quad \text{A.36}$$

D'après l'expression précédente, on constate que les seuils d'appartenance aux prototypes et aux classes ne dépendent pas de la distribution des données. Pour trouver les valeurs $\mu_{\min} = 2$ et $\mu_{\max} = 3$, les constantes k_1 et k_2 sont fixées à $k_1 = -1,4$ et $k_2 = -2,2$.

A.4.3 Le seuil de cardinalité

La notion de prototype élémentaire P_e est introduite pour désigner le plus petit prototype représentatif. Le seuil de cardinalité $N_{\min}$ est introduit pour détecter les prototypes représentatifs. Elle correspond au nombre de données minimal pour estimer correctement le modèle, on a : $N_{\min} = \text{card}(P_e)$. On constate que la cardinalité du prototype élémentaire doit dépendre de la dimension D de données pour que le modèle des prototypes soit définit correctement. Par exemple en 2D, elle caractérise une ellipse; une seule donnée n'est donc pas suffisante pour déterminer sa géométrie. De même en 3D, il faudrait plus que deux points pour déterminer l'ellipsoïde caractérisant le prototype. Dans cette section, nous proposons quelques conditions pour éviter un mauvais choix de seuil $N_{\min}$. Ces conditions imposent les bornes inférieures du seuil de cardinalité $N_{\min}$ suivant les relations :

Si $D = 1$, $\quad \inf(N_{\min}) = 2$

Si $D = 2$, $\quad \inf(N_{\min}) = 3$

⋮

Si $D = n$, $\quad \inf(N_{\min}) = n + 1$

Résumé : La plupart des processus naturels ou artificiels ont des comportements évolutifs décrits par des données non-stationnaires. La problématique étudiée dans cette thèse concerne la classification dynamique de données non-stationnaires. Nous proposons une description générique de classifieurs dynamiques conçue à l'aide d'un réseau neuronal à architecture évolutive. Elle est élaborée en quatre procédures d'apprentissage : création, adaptation, fusion, et évaluation. Deux algorithmes sont développés à partir de cette description générique. Le premier est une nouvelle version de l'algorithme AUDyC (AUto-adaptive and Dynamical Clustering). Il utilise un modèle de mélange décrit suivant l'approche multimodale. Le second, nommé SAKM (Self-Adaptive Kernel Machine), est basé sur les SVM et méthodes à noyau. Ces deux algorithmes sont dotés de règles de mise à jour récursives permettant la modélisation adaptative et le suivi de classes évolutives. Ils disposent de capacités d'auto-adaptation en environnement dynamique et de bonnes performances en terme de convergence et de complexité algorithmique. Ces dernières sont prouvées théoriquement et montrées par la simulation des algorithmes.

Mots clés : apprentissage en ligne, classification automatique, données non-stationnaires, classes évolutives, architecture neuronale, modèles de mélange, SVM et méthodes à noyau

----

Abstract : Most of natural or artificial processes have evolutionary behaviours described by non-stationary data. This thesis studies the problem of dynamical clustering of non-stationary data. We propose a generic description of dynamical classifiers by using a neural network with evolutionary architecture. It is composed of four learning procedures: creation, adaptation, fusion, and evaluation. From this generic description, we develop two algorithms. The first one is a new version of AUDyC (AUto-adaptive and Dynamical Clustering). AUDyC uses a mixture model described following the multimodal approach. The second one, called SAKM (Self-Adaptive Kernel Machine), is based on SVM & kernel methods. Both are created with some recursive update rules that allow the adaptive modelling and the pursuit (tracking) of evolving clusters. They have self-adaptive abilities in non-stationary environment and good performances of convergence and complexity. These latter are theoretically proved and also illustrated by simulation.

Key words : on-line learning, clustering, non-stationary data, evolving clusters, neural architecture, mixture models, SVM and kernel methods