N° attribué par la bibliothèque

Université Paris-Nord

THESE

présentée par

Farida Zehraoui

pour obtenir le titre de

Docteur d'Université

Spécialité Informatique

Sujet:

Systèmes d'apprentissage connexionnistes et raisonnement à partir de cas pour la classification et le classement de séquences

Soutenue publiquement le 12 Octobre 2004
devant le jury composé de

M. Alain	Mille	Rapporteur
M. Nicolas	Nicoloyannis	Rapporteur
M. Younès	Bennani	Directeur
M. Philippe	Dague	Examinateur
M. Maurice	Milgram	Président
Mme Sylvie	Salotti	Examinatrice
M. Françoise	Fessant	Invitée
M. Eric	Janvier	Invité
M. Rushed	Kanawati	Invité

Résumé

La thèse porte sur l'utilisation de techniques d'apprentissage automatique pour le traitement de séquences. Ce travail est motivé par une application réelle qui consiste à modéliser le comportement d'un internaute à partir de traces de navigations dans le but d'adapter de façon dynamique un site Web de commerce électronique. Nous avons d'abord proposé un système de raisonnement à partir de cas (RàPC) « CASEP » pour le classement ou la prédiction à partir de séquences. Des mesures de maintenance sont associées aux cas pour faire face au bruit contenu dans les données et pour effectuer la réduction de la base de cas. Nous avons ensuite proposé plusieurs modèles de cartes auto-organisatrices (SOM) pour la classification et le classement de séquences. Nous nous sommes intéressés à l'insertion des propriétés de plasticité et de stabilité dans une carte SOM ainsi qu'au traitement des séquences temporelles. A la fin de cette étude, nous avons travaillé sur la possibilité de coopération des systèmes connexionnistes avec l'approche de RàPC afin de concevoir des systèmes hybrides. Ces architectures présentent l'avantage de combiner les points forts de leurs différents composants. Nous avons ainsi proposé le système hybride « CASEP2 » qui combine le raisonnement à partir de cas avec l'un des réseaux de neurones que nous avons proposés.

Mots-clés: Classement et classification de séquences, Carte auto-organisatrice, Plasticité et stabilité, Raisonnement à partir de cas, maintenance de la base de cas.

Abstract

The thesis relates to the use of automatic learning techniques for the sequence processing. This work is motivated by a real application, which consists in modelling an e-commerce site user behaviour that aims to adapt dynamically the Web site. First, we have proposed a case based reasoning (CBR) system for the sequence classification or prediction "CASEP". New maintenance measurements are associated to each case in order to take into account the presence of noise in the data and to reduce the case base size. Then, we have proposed several models of self-organizing maps (SOM) for the sequence clustering and classification. In these models, we have introduced the plasticity and the stability properties in the SOM and we have proposed models that process sequences. Finally, we have studied the possibility of the co-operation between the connectionist models and the case based reasoning in order to conceive modular hybrid systems. These architectures integrate theirs modules capabilities. We have thus proposed the hybrid system "CASEP2", which combines the CBR and one of the proposed SOM models.

Keywords: Sequence classification and clustering, Self organizing map, Plasticity and stability, Case based reasoning, Case base maintenance.

Remerciements

La thèse est un travail de longue haleine, qui nécessite l'aide et le soutien de plusieurs personnes. Je souhaiterais donc leur consacrer ces quelques lignes de mon manuscrit pour leur témoigner ma reconnaissance.

J'adresse mes sincères remerciements à Younès Bennani, Rushed Kanawati et Sylvie Salotti qui m'ont permis de réaliser cette thèse, pour leur disponibilité, pour leurs conseils, pour toutes les discussions enrichissantes que nous avons eues ensemble et pour la confiance qu'ils m'ont accordée durant toutes ces années.

Je voudrais remercier Alain Mille pour avoir accepté d'être rapporteur de ma thèse et pour ses remarques pertinentes qui m'ont permis d'améliorer mon manuscrit.

Je tiens à remercier Nicolas Nocoloyannis d'avoir accepté d'être rapporteur de ma thèse malgré la charge que cette fonction représente.

Je remercie Maurice Milgram d'avoir accepté de présider mon jury de thèse.

Je tiens à remercier Phillipe Dague d'avoir accepté d'examiner ma thèse, de m'avoir accueilli au sein de son équipe de recherche (ADAGE) et d'avoir été disponible pour tous les thésards afin qu'ils préparent leurs thèses dans de bonnes conditions.

Je remercie Eric Janvier pour son suivi, pour sa rigueur de travail qui ont contribué au bon déroulement de la coopération avec l'entreprise et pour avoir accepté de faire partie de mon jury de thèse.

Je tiens à remercier Françoise Fessant pour sa participation à mon jury de thèse.

Je remercie Maria Malek et Sandrine Fabre pour leur collaboration et pour les discussions intéressantes que nous avons eues.

Je remercie Sophie Aubin, Thierry Poibeau et Patrick Baillot pour la lecture de certains chapitres de ma thèse.

Je remercie tous les anciens et actuels thésards et ingénieurs du LIPN pour avoir su créer un groupe sympathique et agréable et particulièrement : Touria Ait El Mekki, Sauphie Aubin, Zeina Jrad, Hakima Kadri Dahmani, Hager Karoui, Hassine Moungla, Farid Nouioua, Gabriel Paillard, Babacar Thiongiane et Guillaume Vauvert.

La réalisation de ce travail s'appuie également sur un environnement qui est essentiel. Je tiens à remercier tout les membres du laboratoire LIPN pour la bonne ambiance qui y règne, pour leur soutien, leurs conseils et les discussions enrichissantes que j'ai pu avoir avec eux.

Ma plus profonde gratitude va à mes parents. Tout au long de mes études, ils m'ont soutenue encouragée et aidée pour me donner toutes les chances de réussir. Qu'ils trouvent dans la réalisation de ce travail, l'aboutissement de leurs efforts ainsi que l'expression de ma très grande reconnaissance.

Je remercie mes frères et sœurs pour leur soutien moral et leur réconfort dans les moments difficiles ainsi que pour leur encouragement malgré la distance qui nous sépare.

Je remercie mes oncles, ma tante et mes cousines résidant en France pour leur soutien depuis mon arrivée en France.

Je remercie également tous ceux que j'ai oubliés de citer et dont j'ai croisé la route d'une façon ou d'une autre au cours de ces années riches d'expérience.

à ma mère et à mon père

Table des matières

Chapitre 1

Introduction

Notre travail porte sur l'utilisation des systèmes d'apprentissage connexionnistes et de la méthodologie du raisonnement à partir de cas (RàPC) pour le traitement de séquences dans un contexte opérationnel pour une classe de problèmes complexes.
L'étude est réalisée dans le cadre du projet « MASC »[1] (Modélisation et Apprentissage Symbolique et Connexionniste de comportements utilisateurs) dont l'objectif est d'étudier et de développer des méthodes d'apprentissage automatique de comportements d'utilisateurs à partir de traces de navigations sur le Web. Nos contributions sont validées par le développement d'une application concrète qui consiste à modéliser le comportement d'un internaute à partir de traces de navigations dans le but d'adapter de façon dynamique un site Web de commerce électronique.

1.1 Contexte

Avec la compétition inter-entreprises qui ne cesse de croître, augmenter les ventes d'une entreprise devient un besoin vital. En effet, avec la mondialisation des marchés et l'abolissement des frontières commerciales, le marché local devient souvent insuffisant pour la survie des entreprises. Le commerce électronique est l'un des moyens qui permet à l'entreprise qui le pratique de commercialiser ses produits dans le monde entier en utilisant le Word Wide Web (Web).
Or, quand une personne visite un site de commerce électronique, elle peut facilement être frustrée du fait qu'elle met beaucoup de temps pour trouver les produits qui satisfont ses besoins ou du fait que les informations fournies sur ces produits sont très techniques et difficiles à comprendre ou au contraire sont trop superficielles et ne sont pas assez détaillées. Pour faire face à ce problème, un site de commerce électronique doit être conçu de façon à s'adapter aux besoins de ses utilisateurs.
Pour ceci, la plupart des travaux dans ce domaine se sont intéressés à modéliser le comportement des utilisateurs du site afin de parvenir à satisfaire leurs besoins.
Pour modéliser le comportement de l'utilisateur, deux types d'informations (données) peuvent

[1] C'est un projet entre le laboratoire LIPN et la société NUMSIGHT CONSULTING.

être disponibles dans le site :

- *Données explicites :* elles sont fournies par l'utilisateur par le biais de questionnaires (par exemple son age, sa profession, etc.)
- *Données comportementales :* elles sont fournies par l'internaute de façon implicite à travers sa navigation dans le site (par exemple les pages visualisées, le chemin parcouru dans le site, les produits achetés, etc.)

Les fichiers de traces de navigations (fichiers log), qui contiennent des données d'usage, représentent l'une des plus importantes sources d'informations sur la navigation dans un site Web puisqu'elles sont les plus disponibles et les moins coûteuses.
Il existe plusieurs formats de représentation d'un fichier log. Le format standard utilisé par les serveurs qui enregistrent les accès effectués par les navigateurs est la "COmmon Log Application" créée par le NCSA (National Center for Supercomputing Applications) [*http ://www-3.ibm.com/software/webservers/siteanalyzer/doc/v41/Infocenter/guide/c-logs.html*].

Time	c-ip	cs-username	cs-method	cs-uri-stem	sc-status	sc-bytes	cs(User-Agent)	Cs(Referer)
00:01:39	213.36.9.210	-	GET	/index.asp	200	245	Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98)	http://www.ifrance.com/boutique/
00:01:54	213.36.9.210	-	GET	/Navigation/searchack.asp	200	283	Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98)	-
00:01:54	213.36.9.210	-	GET	/images/BkgdMmain2.gif	200	300	Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98) Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98)	http://www.wstrore.fr/Navigation/searchack.asp?action=search%2FperCategory%2Easp&cat_id=524
00:02:00	213.36.9.210	-	GET	/images/FileGreenL.gif	200	234	Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98)	http://www.wstrore.fr/Navigation/searchack.asp?action=search%2FperCategory%2Easp&cat_id=524
00:02:02	213.36.9.210	-	GET	/images/FileGreenR.gif	200	400	Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98)	http://www.wstrore.fr/Navigation/searchack.asp?action=search%2FperCategory%2Easp&cat_id=524
00:02:02	213.36.9.210	-	GET	/images/BkgdFileGreen.gif	200	420	Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98)	http://www.wstrore.fr/Navigation/searchack.asp?action=search%2FperCategory%2Easp&cat_id=524
00:01:58	213.36.9.210	-	GET	/search/perCategory.asp	200	200	Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98)	-

FIG. *1.1: Exemple d'un fichier log standard.*

Un fichier log standard (figure 1.1) fournit généralement les informations suivantes :

- L'adresse IP de l'utilisateur

- L'identificateur de l'utilisateur « username » : C'est un attribut décrivant l'identité de l'utilisateur. La plupart du temps, cet attribut n'est pas fourni
- La date et l'heure de la requête
- L'adresse de la page Web accédée (l'URL)
- La taille du fichier reçu : elle représente le nombre total d'octets transférés par le serveur au client durant la transaction
- Les méthodes utilisées pour la requête
- Le statut : décrit l'état de la transaction (succès/échec)
- La version du navigateur utilisée par l'internaute

Pour notre application, nous disposons d'un grand volume de données. Ces données sont des séquences de longueur variable (selon le nombre de pages parcourues par l'utilisateur du site) fournies par les fichiers log.
A la demande du partenaire industriel, nous nous sommes intéressés à la classification des internautes selon leur comportement et à leur classement dans l'une des deux classes : {acheteur, non acheteur}. Ceci doit être effectué le plus vite possible afin de pouvoir effectuer des actions durant les navigations des utilisateurs. De plus, nous ne disposons pas de connaissances *a priori* du domaine. Les actions qui seront ensuite effectuées peuvent être : « proposer des pages publicitaires pour les non acheteurs », « proposer des produits qui pourraient intéresser les acheteurs », etc.

1.2 Codage des données

Comme l'utilisation directe des données brutes contenues dans un fichier log n'est pas possible, plusieurs traitements ont été effectués avant d'utiliser les différents modèles (voir la figure 1.4).

Les données sont d'abord codées en utilisant des matrices quasi-comportementales (Zeboulon *et al.*, 2003) après différents filtrages qui ont permis de supprimer une partie du bruit.
Le principe de ce codage, consiste à calculer pour chaque page sa fréquence de précédence et de succession sur toutes les autres pages (voir le tableau de la figure 1.2). L'idée intuitive est que deux pages sont similaires si elles sont en général précédées par les mêmes pages et suivies par les mêmes pages dans la navigation.

La matrice quasi-comportementale est une méthode qui a été élaborée à cause de la non disponibilité des variables qui devraient caractériser les URLs. Cette méthode consiste donc à calculer la matrice de précédence et de succession sur la totalité du fichier Log. Grâce à ce codage, on a réussit à caractériser chaque page du site selon sa présence dans les navigations enregistrées dans le fichier. Ce codage tel qu'il était décrit pose un problème d'effectif. En effet, n'ayant pas

P	**Entrée**	**P_1……...P_j ……..…P_N**	**P_1.....P_k……..…P_N**	**Sortie**
Entree	MaxE	0 0 0 ……...…..…0		0
P_1 P_i P_N	a	b	c	d
Sortie	0		0 0 0 …..……..0	MaxS

Précédence Succession

FIG. *1.2: Codage par matrice quasi-comportementale. MaxE représente le nombre de pages apparues comme entrée,MaxS est le nombre de page apparues comme sortie, P_iPP_j veut direP$_i$ précède P_j et P_iSP_j signifie P_i succède P_j.*

			Précède	Suit	
Individus(URLs)	jour	E	URL_1 URL_2 URL_j .... URL_N	URL_1 URL_2 URL_l ...URL_N	S
E URL_1 URL_2 URL_i URL_N S E URL_1 URL_2 URL_i URL_N S					
… E URL_1 URL_2 URL_i URL_N S	*k*	*a*	..... *b*	..... *c*	*d*

FIG. *1.3: Tableau de codage en utilisant la matrice quasi-comportementale dynamique. Dans le k^{eme} jour du fichier l' URL_i est apparue a fois comme page d'entrée, précédée b fois par la page URL_j, suivie c fois par la page URL_l et apparue d fois comme page de sortie.*

assez d'URLs significatives dans le site, on ne peut avoir un tableau avec un nombre de données suffisant afin de construire un modèle robuste.

Pour remédier à ce problème, les auteurs de (Benabdeslem *et al.*, 2002) ont proposé de glisser la matrice sur le mois. En d'autres terme, ils ont calculé des matrices quasi-comportementales par jour, voire même par semaine ou généralement par pourcentage sur la base. Cette méthode, que nous avons utilisée, nous permet de multiplier le nombre d'échantillons et d'avoir plusieurs exemples pour chaque URL (voir le tableau de la figure 1.3).

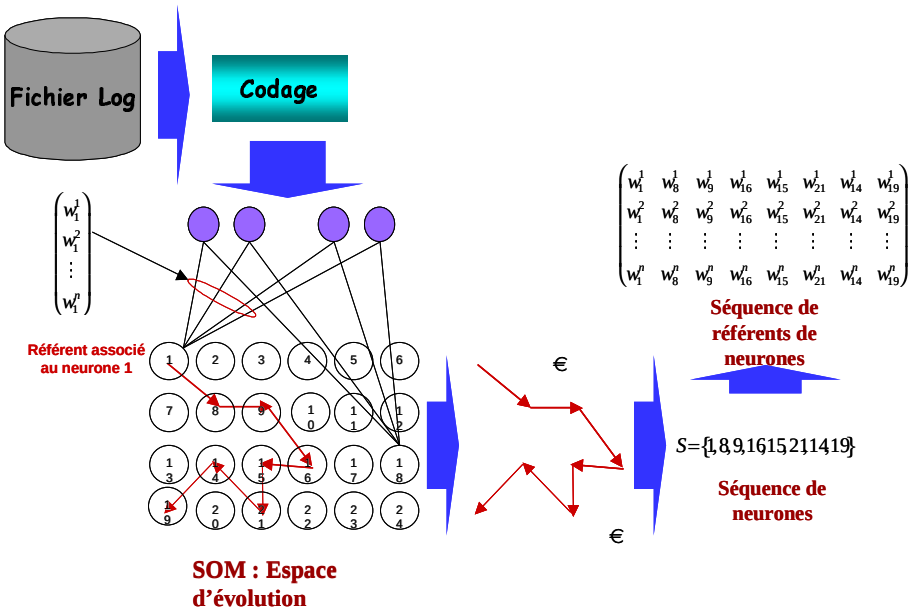


FIG. *1.4: Les différentes étapes effectuées pour le codage des données avant leur utilisation.*

Malgré ce codage, les matrices quasi-comportementales restent complexes et difficiles à interpréter. Pour résumer l'information contenue dans ces matrices de manière simple, les données codées ont été traitées par une carte auto-organisatrice. Celle-ci permet de définir un espace d'évolution dans la carte. Chaque neurone de cette carte est un représentant d'un ensemble de pages similaires. Les navigations des internautes sont alors représentées par les trajectoires données par les successions des neurones. Comme chaque neurone est caractérisé par son vecteur poids (ou référent), la navigation est représentée par la succession des ces vecteurs (voir la figure 1.4).

1.3 Problématique

A partir de l'application décrite dans le paragraphe précédent (§1.1), nous avons extrait plusieurs caractéristiques pertinentes sur lesquelles nous nous sommes basés dans notre travail. Ces caractéristiques sont :

- Un volume très important de données ;
- Des données dynamiques ;
- Des données constituées de séquences d'événements ;
- Des contraintes temps réel ;
- Manque de connaissances *a priori* du domaine ;

– Présence de bruit dans les données.

En effet, le comportement de chaque utilisateur est décrit par la succession de pages qu'il a parcourues. Le site de e-commerce est consulté chaque jour par des milliers de personnes. Par conséquent, nous avons un grand volume de données qui sont dynamiques puisqu'elles peuvent constamment changer avec le lancement de nouveaux produits sur le marché et la suppression d'autres. Comme notre travail est réalisé pour pouvoir effectuer des actions durant la navigation de l'internaute, les classifications et les classements (ou les prédictions) doivent être effectuées avant qu'il ne quitte le site, ce qui impose des contraintes temps réel. Dans notre application, nous possédons peu de connaissances *a priori* du domaine et nous pensons qu'il est nécessaire d'utiliser les expériences passées fournies par d'autres internautes qui ont navigué dans le site. Ces expériences sont extraites à partir des données brutes collectées qui sont contenues dans les fichiers log et qui contiennent beaucoup de bruit.
Notre but est de proposer des approches pour toute une classe de problèmes possédant les mêmes caractéristiques que notre application. Cette classe de problèmes peut inclure plusieurs autres applications. Dans le domaine de la robotique, par exemple, la prédiction des actions futures d'un robot mobile dépend de la succession des actions qu'il a effectuées lors de son interaction avec l'environnement. Les actions que le robot doit effectuer doivent être prises en temps réel pour éviter les accidents. Dans le domaine de l'apprentissage assisté par ordinateur, les apprenants peuvent être modélisés par la succession des documents qu'ils consultent, afin de pouvoir les former selon leurs besoins et à leur propre rythme au travers d'une interaction en temps réel avec l'ordinateur.
Le *traitement des séquences temporelles* consiste à effectuer l'une des tâches suivantes : la classification, le classement ou la prédiction.

– La classification consiste à former des groupes constitués de séquences similaires qui représentent des comportements proches.
– Dans la tâche de classement, une classe est associée à une séquence d'entrée. Cette classe est choisie parmi les classes des séquences stockées dans la mémoire.
– Pour la prédiction, une valeur future d'une séquence donnée est estimée en se basant sur ses valeurs passées observées.

Pour effectuer ces tâches, nous nous sommes intéressés à la méthodologie de raisonnement à partir de cas ainsi qu'aux techniques connexionnistes connues pour leur efficacité dans le traitement de gros volumes de données. De plus, le raisonnement à partir de cas et les réseaux connexionnistes peuvent être utilisés quand les connaissances *a priori* du domaine ne sont pas disponibles.

1.4 Présentation du raisonnement à partir de cas

Le raisonnement à partir de cas (RàPC) (Aamodt et Plaza, 1994) est une méthodologie de résolution de problèmes basée sur la réutilisation d'expériences passées. Ces expériences qui servent à résoudre de nouveaux problèmes sont constituées des problèmes déjà résolus appelés *cas sources*. Un cas est généralement composé de deux parties décrivant un problème et la solution qui lui est appliquée. Lorsqu'un nouveau problème se présente, il est intégré dans un cas, appelé *cas cible*, dont la partie solution est inconnue (elle sera apportée par le raisonnement). Le cas cible est résolu en trouvant un ou plusieurs cas sources qui lui sont similaires et en adaptant leurs solutions à la nouvelle situation. Une autre caractéristique du RàPC est qu'il s'agit d'une approche incrémentale, c'est à dire qu'une nouvelle expérience peut être mémorisée à chaque fois qu'un problème est résolu, ce qui la rend disponible tout de suite après pour la résolution des problèmes futurs.

1.4.1 Structure d'un cas

Un cas est une connaissance qui représente une expérience. Cette expérience constitue une leçon qui permet au système de RàPC d'atteindre ses objectifs. Les informations contenues dans un cas dépendent du domaine d'application et des objectifs pour lesquels ce cas est construit. Par exemple, dans un système de RàPC pour le diagnostic médical, un cas peut représenter l'historique complet de l'état de santé d'un individu ou peut être limité à une seule visite chez le médecin. Dans la dernière situation, le cas peut être un ensemble de symptômes avec le diagnostic et peut aussi inclure le traitement. Si le cas représente l'historique complet d'une personne alors un modèle plus complet est utilisé. Celui-ci peut inclure les changements de symptômes d'une visite à une autre. L'obtention du diagnostic et du traitement est plus difficile que dans la situation d'une simple visite. Mais dans cette dernière, les changements de symptômes, qui peuvent être très utiles pour soigner le patient, sont ignorés.
Un cas est habituellement constitué de deux parties :

- *La partie problème :* Cette partie peut contenir : les objectifs que l'on veut atteindre, les contraintes sur ces objectifs et les attributs qui décrivent le problème ainsi que les relations entre eux.
- *La partie solution :* Cette partie peut regrouper : la description de la solution apportée par le raisonnement, les étapes qui ont mené à cette solution, sa justification, son évaluation et l'effet de son application sur les résultats du système de RàPC.

Le cas source peut aussi contenir une autre partie contenant des informations sur son utilisation dans le système. Cette partie est appelée « information de qualité » (Reinartz *et al.*, 2000).

La base de cas dans un système de RàPC est la mémoire qui contient tous les cas précédemment retenus. Lors de la création d'une base de cas, trois points principaux doivent être

considérés (Main *et al.*, 2000) :

- La structure et la représentation des cas.
- Le modèle de la mémoire utilisée pour organiser la base de cas.
- La sélection des indices qui sont utilisés pour identifier chaque cas.

Afin de permettre la remémoration d'un cas, sa partie problème est décrite par un ensemble de caractéristiques pertinentes appelées *indices*. Ces indices vont déterminer dans quelles situations et dans quels contextes les cas seront remémorés.
Quand un cas est ajouté à la base de cas, le problème qui se pose est de savoir quels indices générer pour ce cas. Le problème du choix des indices dépend du domaine d'application, mais ces indices doivent vérifier certaines propriétés. Il sont construits de façon à être :

- *Prédictifs* afin de jouer un rôle dans la détermination d'une solution pour un nouveau problème.
- *Suffisamment abstraits* de manière qu'un cas puisse être utilisé plusieurs fois pour la résolution de différents problèmes.
- *Suffisamment concrets* afin de pouvoir être reconnus le plus rapidement possible pour la résolution d'un nouveau problème.

1.4.2 Cycle du raisonnement à partir de cas

Le raisonnement à partir de cas est effectué généralement suivant un cycle constitué de quatre phases principales (Aamodt et Plaza, 1994) (voir la Figure 1.5) :

- La recherche
- La réutilisation
- La révision
- L'apprentissage

1. ***Phase de recherche***
   A partir de la description de la partie problème du cas cible, cette phase permet de remémorer un ou plusieurs cas sources à partir de la base de cas. Les méthodes de remémoration dépendent de la façon dont la mémoire est organisée. Il est souhaitable que la remémoration soit la plus correcte possible, c'est à dire que l'expérience rappelée soit, dans la mesure du possible, celle qui permet d'obtenir la meilleure solution à partir des cas présents dans la base de cas. Les tâches effectuées durant cette phase sont : l'identification des caractéristiques pertinentes du problème, la remémoration et la sélection des meilleurs cas parmi les cas sources.
   Le choix des cas remémorés dépend des approches. Certaines sont basées sur des mesures

de similarité entre les cas sources et le cas cible (McSherry, 2002), d'autres utilisent, en plus de la notion de similarité, celle de la diversité (Smyth et McClave, 2001). Le but de ces approches est de remémorer des cas similaires au cas cible et de choisir parmi ces cas ceux qui ne sont pas très similaires entre eux. D'autres approches remémorent des cas adaptables en se basant sur des connaissances d'adaptation (Smyth et Keane, 1995a).

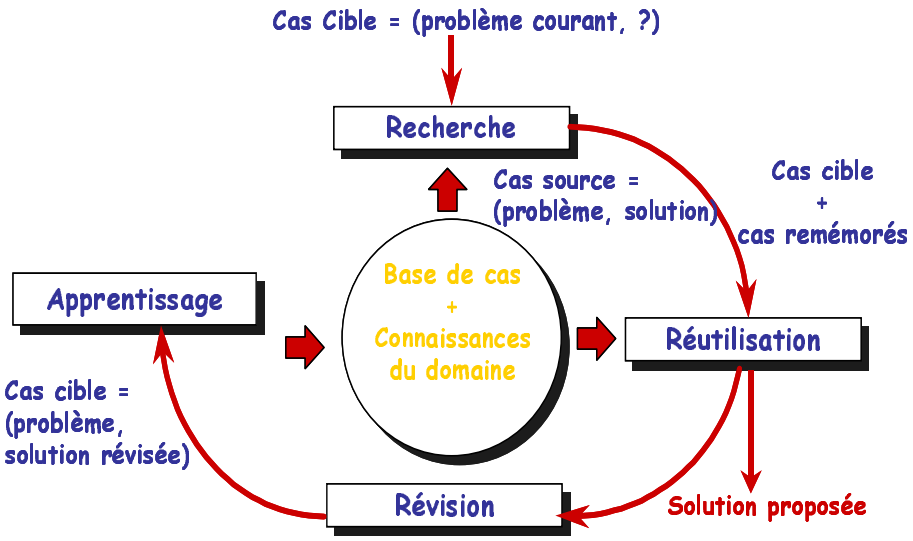


FIG. *1.5: Cycle du raisonnement à partir des cas*

2. ***Phase de réutilisation***

   Cette phase consiste à adapter la solution du cas source remémoré pour résoudre le cas cible. L'adaptation consiste à effectuer un raisonnement par analogie (Lieber *et al.*, 2004) : « *Sachant que la solution du cas cible est à la solution du cas source ce que le cas cible est au cas source, connaissant le cas source et sa solution ainsi que le cas cible, que vaut la solution du cas cible* ? »

   D'après (Fuchs *et al.*, 1999), l'adaptation peut être considérée comme un plan dont l'état initial est la solution de départ et l'état final est la solution adaptée. Le fait de modéliser l'adaptation de cette façon permet d'envisager de façon théorique et pratique la combinaison de l'adaptation avec la remémoration, pour être en mesure de retrouver un cas adaptable lors de la remémoration et de guider l'adaptation d'un tel cas.

3. ***Phase de révision***

   Cette phase consiste à évaluer la solution proposée par l'étape de réutilisation. Si le résultat de la révision est satisfaisant, on passe à l'étape suivante qui est celle d'apprentissage. Par contre, si le résultat de l'évaluation n'est pas bon, la solution est corrigée en utilisant les

connaissances du domaine ou bien en consultant un expert.

4. ***Phase d'apprentissage***
   C'est le processus qui permet d'incorporer ce qui est utile à retenir, à partir de l'étape de résolution d'un problème, dans les bases de connaissances existantes. L'apprentissage peut s'effectuer, par exemple, à partir du succès ou de l'échec dans la résolution du cas cible. Les questions essentielles qui se posent à cette étape sont :

   - Quelle information retenir à partir d'un cas cible résolu ?
   - Sous quelle forme ?
   - Comment indexer ce cas ?
   - Comment l'intégrer dans la structure de la mémoire ?

1.4.3 Pourquoi le RàPC ?

Le choix du RàPC pour notre classe de problèmes est motivé par les aspects suivants (Main *et al.*, 2000) :

L'incrémentalité de l'acquisition des connaissances
Le RàPC effectue un apprentissage incrémental de cas durant son utilisation. Ceci permet à tout nouveau cas résolu, jugé utile pour le raisonnement, de contribuer à la résolution des problèmes futurs. L'ajout de ces cas peut permettre au système de résoudre plus de problèmes et d'améliorer les résultats qu'il fournit.

La possibilité d'effectuer le raisonnement dans un domaine à connaissances réduites
Quand les connaissances du domaine sont insuffisantes pour construire un modèle causal, le RàPC peut fonctionner avec seulement un ensemble de cas du domaine.

La possibilité de raisonner avec des données imprécises ou incomplètes
Comme les cas sources remémorés à partir de la base de cas ne sont pas seulement ceux qui sont identiques au cas cible mais aussi ceux qui lui sont similaires, les mesures de similarité utilisées peuvent tenir compte des imprécisions et des incomplétudes. Ces facteurs peuvent causer, dans certaines situations, une dégradation des résultats du système, mais le raisonnement continue et le système peut apprendre à partir de ces échecs.

Bien que le RàPC soit utilisé dans plusieurs domaines, il y a des situations où ce raisonnement n'est pas approprié. L'application du RàPC repose sur un nombre de caractéristiques des problèmes et de leur domaine. Ces caractéristiques sont utilisées pour déterminer si le RàPC est applicable. Si la réponse à la majorité des questions citées ci-dessous est positive, alors l'appli-

cation du RàPC peut être pertinente pour atteindre l'objectif visé (Main *et al.*, 2000) :

- Est ce que le domaine possède un modèle sous-jacent ?
  Si le processus est aléatoire, ou si les facteurs qui ont mené au succès ou à l'échec d'une solution ne peuvent pas être pris en compte dans la description du cas, le raisonnement à partir de cas peut ne pas être approprié.
- Y a-t-il des exceptions et de nouveaux cas ?
  Les domaines qui ne possèdent pas de nouveaux cas ou des exceptions peuvent être mieux modélisés par des règles.
- Est ce que les cas se reproduisent ?
  Dans les domaines où les cas ne sont pas suffisamment similaires aux nouveaux problèmes, qui se présentent, pour être adaptés, alors il est peut être meilleur d'utiliser un modèle qui construise entièrement les solutions qu'un modèle basé sur la réutilisation de solutions.
- Y a-t-il un bénéfice significatif dans l'adaptation des solutions des anciens cas ?
  Avant de concevoir un système de RàPC, il faut considérer s'il y a une différence significative dans les ressources utilisées (temps, traitement, etc.), entre la construction complète d'une solution au problème et l'obtention de la solution par l'adaptation d'une solution similaire.

Notre problème nécessite d'effectuer des classifications, classements (ou prédictions) à partir de séquences dans un domaine où les connaissances *a priori* du domaine ne sont pas disponibles, tout en ayant à notre disposition un gros volume de données brutes. Concernant les conditions d'utilisation du RàPC, notre approche soutient l'hypothèse suivante[2] : *la majorité des utilisateurs du site ayant des comportements proches ont des buts proches (sans exclure l'existence de certaines exceptions qui doivent être prises en compte par notre système).* Le comportement des internautes n'est alors pas un processus aléatoire. De plus, plusieurs utilisateurs peuvent faire des parcours qui se ressemblent dans le site, ce qui fait que des anciens cas peuvent être réutilisés pour résoudre de nouveaux cas. Comme l'environnement est dynamique, de nouveaux cas peuvent se présenter et peuvent être ajoutés à la base de cas. Quant à la réutilisation des solutions fournies par d'autres utilisateurs, nous pensons qu'elle est nécessaire puisqu'il est difficile de détecter la classe de l'internaute sachant que nous n'avons pas de connaissances *a priori* du domaine.
D'après les arguments que nous venons de citer, nous pensons que le RàPC est une méthodologie appropriée pour atteindre notre objectif.

Cependant, l'utilisation du raisonnement à partir de cas pour notre classe de problèmes complexe peut poser un certain nombre de difficultés liées à l'aspect séquentiel et à la présence du bruit ainsi qu'au gros volume de données et aux contraintes temps réel. Ceci sera discuté dans le paragraphe suivant.

[2]Cette hypothèse a été utilisée dans plusieurs travaux comme (Jaczynski et Trousse, 1998),(Kanawati *et al.*, 1999) et a été soutenue par les experts du domaine.

1.4.4 L'utilisation du RàPC pour notre classe de problèmes

L'utilisation des données constituées de séquences peut entraîner certaines difficultés qui ne se posent pas lorsqu'on travaille seulement avec des données « attributs-valeurs ». Ces difficultés concernent d'abord la représentation du cas dans le système de RàPC.

Nous pouvons citer en premier le choix entre la représentation de cas sous forme d'une succession d'évènements instantanés (approche à base de points ou d'instants) et la représentation du cas sous forme de relations entre des intervalles temporels (approche à base d'intervalles ou de périodes) (Jaere *et al.*, 2002). Dans quel cas choisir l'une ou l'autre de ces représentations ? La première représentation est la plus utilisée dans la littérature (Jaczynski, 1998), (Jaczynski et Trousse, 1998), (Fuchs *et al.*, 1995), (Rougegrez-Loriette, 1994), (Malek et Kanawati, 2001) et (Corvaisier *et al.*, 1997).

Comme nous avons des données brutes constituées de séquences de longueurs différentes, le choix de la granularité du cas est nécessaire. Est-ce qu'un cas va représenter l'ensemble de la séquence ou bien juste une expérience précise dans la séquence ? L'idéal est que le cas soit représenté par un extrait pertinent de la séquence qui permette de la caractériser pour un objectif donné. Mais comment retrouver cet extrait dans une séquence temporelle ?
Certaines heuristiques ont été proposées dans la littérature (Jaczynski et Trousse, 1998), (Kanawati *et al.*, 1999) et (Malek et Kanawati, 2001), mais il n'y a pas de méthode générale qui permette l'extraction des évènements pertinents dans une séquence.
Quand le cas est une expérience précise dans une séquence (sous-séquence), comment organiser la mémoire du système ? Est-ce que cette mémoire va contenir seulement les séquences entières ? Ou bien va t-elle contenir seulement les cas cibles résolus ? Il y a aussi une autre alternative qui consiste à garder les séquences entières ainsi que les cas dans la mémoire du système. Cette alternative est certes plus coûteuse en espace mémoire, mais peut améliorer les résultats du système. Le choix de la structure de données à utiliser pour mémoriser les séquences entières et/ou les sous-séquences (cas) est aussi important lors de la conception d'un système de RàPC.

Comme les cas peuvent être de longueurs différentes, ceci rend la tâche de recherche de cas similaires au cas cible plus complexe. Quel algorithme utiliser pour tenir compte de ce problème ? Il existe des algorithmes permettant d'aligner deux séquences de façon à avoir une similarité maximale (Needleman et Wunsch, 1970) entre ces séquences. Mais ces algorithmes sont-ils pertinents pour tout objectif d'un système de RàPC ? Ou bien faut-il procéder d'une autre manière pour calculer les similarités ?

Quand la mémoire du système contient les séquences entières, comment extraire les cas sources à partir de ces séquences ? Est ce que ces cas extraits auront exactement la même succession d'évènements que dans le cas cible ? Ou bien va-t-on tolérer des différences entre ces cas (comme par

exemple, des écarts entre deux événements consécutifs) ?
Comment organiser la recherche dans la mémoire du système de RàPC quand elle contient les séquences entières et les cas (sous-séquences) ?

Lors de l'utilisation du système de RàPC pour le classement (ou la prédiction) de séquences, les évènements arrivent au fur et à mesure dans le temps. Sachant que dans une même séquence, à l'arrivée de chaque évènement, nous pouvons construire un nouveau cas cible, comment tirer profit des raisonnements précédents pour résoudre les nouveaux cas cibles sachant que les cas d'une même séquence peuvent contenir plusieurs informations communes ?

Plusieurs approches, ayant traité des séquences à l'aide du RàPC, ont proposé des solutions et/ou heuristiques pour résoudre certains problèmes liés à la représentation des cas sous forme de séquences d'événements. Ceci sera discuté dans le chapitre suivant (chapitre 2).

Le temps mis pour remémorer les cas dépend fortement de la représentation du cas et de l'organisation de la mémoire du système. Lorsque le cas représente toute la séquence, la recherche de cas similaires peut nécessiter beaucoup de temps quand la séquence est très longue. Le choix d'une structure de données adaptée pour la base de cas peut réduire ce temps mais cette réduction est-elle suffisante pour satisfaire les contraintes temps réel ? Lorsque le cas représente une expérience précise dans la séquence, la phase d'élaboration du cas doit être effectuée en un temps acceptable. Ensuite, il y a la phase de recherche qui diffère selon le cas où la mémoire contient juste les séquences entières, juste les cas (sous-séquences) ou bien les deux.
Dans la première situation, les cas sources sont extraits à chaque fois à partir des séquences entières. Cette extraction nécessite parfois la comparaison de tous les états contenus dans le cas cible avec ceux contenus dans toutes les séquences, ceci peut être très coûteux en temps de calcul. Comme précédemment, le choix d'une structure de données adéquate peut réduire le temps de calcul, mais pas nécessairement de manière à satisfaire les contraintes temps réel.
Dans la deuxième situation, seuls les cas (sous-séquences) sont gardés en mémoire. Cette approche semble moins coûteuse en temps mais les résultats du système peuvent ne pas être satisfaisants à cause de l'absence des séquences entières qui peuvent contenir des informations plus complètes.
Dans la troisième situation, les cas et les séquences entières sont mémorisés. Le temps mis dans la remémoration de cas sources dépend de la manière dont la recherche de cas similaires au cas cible est effectuée. Les résultats fournis par le système peuvent être meilleurs que dans les deux situations précédentes.

Une bonne indexation de cas dans un système de raisonnement à partir de cas qui effectue une partition de la base de cas peut permettre de réduire le temps de recherche des cas sources. Pour notre classe de problèmes, l'indexation doit tenir compte du fait que les cas soient des séquences temporelles. Plusieurs méthodes d'indexation de cas existent dans la littérature (Jaczynski, 1998) et (Malek, 1996). Nous nous intéressons particulièrement aux méthodes connexion-

nistes d'indexation. Les méthodes existantes pour indexer les cas sont destinées aux cas « attribut-valeurs » : les réseaux de neurones utilisés sont des réseaux classiques qui ne sont pas adaptés aux traitement des séquences. Comment choisir un réseau de neurones qui prenne en compte l'information temporelle contenue dans la séquence ? Sachant qu'un tel réseau doit satisfaire certaines conditions liées aux caractéristiques de notre classe de problèmes. Ces conditions portent sur l'utilisation à long terme du système dans un environnement dynamique où les données changent constamment. Ce réseau de neurones doit être capable d'acquérir de nouvelles connaissances sans détruire les anciennes.

La remémoration du cas le plus similaire au cas cible pour proposer une solution ne peut être fiable à cause de la présence de bruit. Donc un ensemble de cas doit être remémoré pour fournir une solution au cas cible. Comment fournir cette solution sachant que les connaissances d'adaptation ne sont pas disponibles ? L'utilisation d'un réseau de neurones peut-elle pallier à ce problème ?

Dans toutes les situations que nous avons citées, la base de cas d'un système RàPC peut être volumineuse et les cas sources peuvent contenir du bruit. Ceci doit être pris en compte pour arriver à satisfaire les contraintes temps réel et à construire un système qui fournisse des résultats satisfaisants. La maintenance d'un système de raisonnement à partir de cas (Leake et Wilson, 1998) s'avère donc nécessaire.

En plus de la méthodologie du RàPC, nous nous sommes intéressés dans notre étude à l'utilisation des systèmes d'apprentissage connexionnistes. Nous allons présenter, dans le paragraphe suivant, les modèles connexionnistes (réseaux de neurones artificiels) ainsi que l'intérêt de leur utilisation pour notre classe de problèmes.

1.5 Systèmes d'apprentissage connexionnistes

Un réseau de neurones artificiels peut être défini comme un ensemble d'unités de calcul reliées entre elles par des liens appelés « connexions ». La plupart des réseaux de neurones font appel à des règles d'apprentissage à partir de données pour ajuster les poids des connexions. En d'autres termes, les réseaux de neurones sont généralement élaborés à partir d'exemples. Ils représentent ensuite une certaine capacité de généralisation pour des données non présentes dans la base d'apprentissage.

1.5.1 Modèle d'un neurone formel

Il existe un grand nombre de modèles de neurones. Les modèles les plus utilisés sont basés sur le modèle développé par McCulloch et Pitts (McCulloch et Pitts, 1943). Le neurone peut être représenté par un processeur possédant plusieurs entrées et une sortie et peut être modélisé

par deux opérateurs (voir la Figure 1.6) :

- Un opérateur de sommation I égal à la somme pondérée des entrées x_i $(1 \leq i \leq n)$ et des poids w_i $(1 \leq i \leq n)$ du neurone : $I = \sum_{i=1}^{n} w_i x_i$.
- Un opérateur de décision qui calcule l'état de la sortie y du neurone en fonction de son potentiel I : cet opérateur est appelé « fonction de transition » : $y = f(I)$.

Dans le cas du modèle de McCulloch et Pitts (McCulloch et Pitts, 1943), l'opérateur de décision est une fonction à seuil. Le neurone est caractérisé par deux états ; sa valeur dépend du potentiel I et d'un seuil θ. Ce genre de neurones est appelé « neurone binaire ».
L'utilisation d'une fonction non dérivable (comme la fonction à seuil) comporte quelques inconvénients lors de l'apprentissage (Jodouin, 1994). On fait souvent appel à des fonctions monotones, croissantes et dérivables. L'état du neurone est alors multivalué. Pour effectuer un classement, des fonctions de type « sigmoïde » sont souvent utilisées, comme par exemple une tangente hyperbolique dont la sortie est bornée entre -1 et 1.

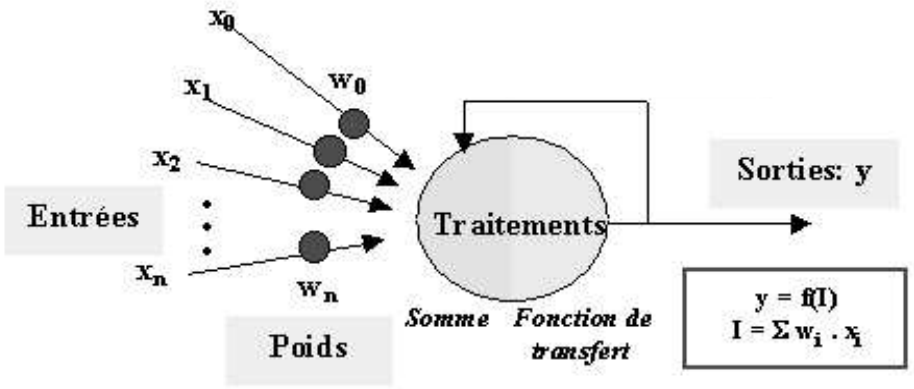


FIG. *1.6: Modèle du neurone.*

1.5.2 Réseaux de neurones artificiels

Dans un réseau de neurones artificiels, trois types de neurones peuvent être distingués (voir la figure 1.7) :

- Les neurones d'entrée, aussi appelés cellules perceptives puisqu'elle acquièrent des données dont la provenance est en dehors du réseau ;
- Les neurones de sortie, qui définissent la réponse du réseau ;

– Les neurones cachés, qui n'ont aucune relation avec le monde extérieur au réseau, ils communiquent juste avec les autres neurones du réseau.

Architecture

La plupart des réseaux de neurones ont une topologie définie sous forme de couches. Il existe quelques exceptions lorsque les réseaux ne sont pas explicitement définis sur plusieurs couches, mais ils peuvent être considérés comme n'ayant qu'une seule couche. L'architecture du réseau peut alors être décrite par le nombre de couches et le nombre de neurones dans chaque couche.

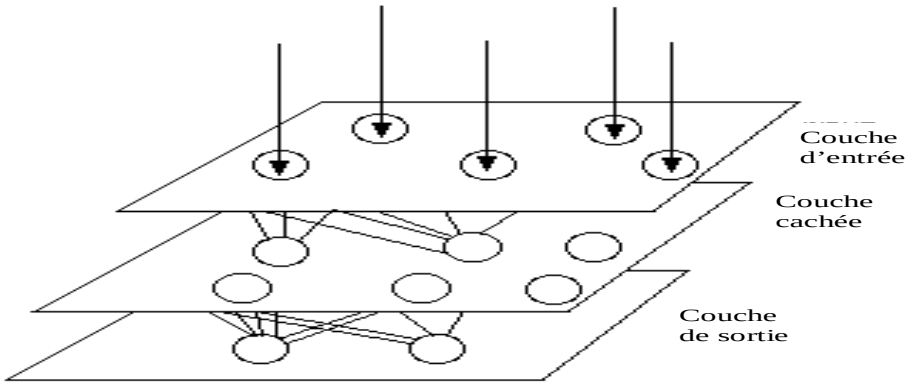
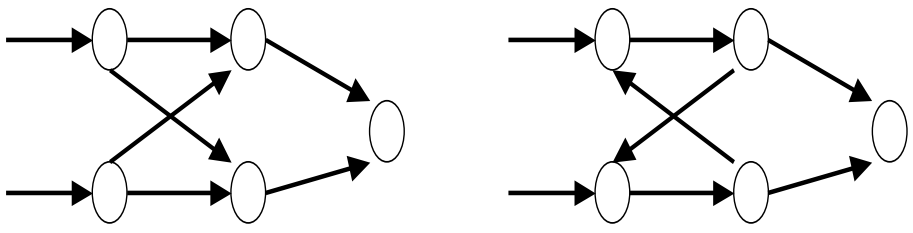


FIG. *1.7: Architecture d'un réseau de neurones multi-couches.*

Connexions

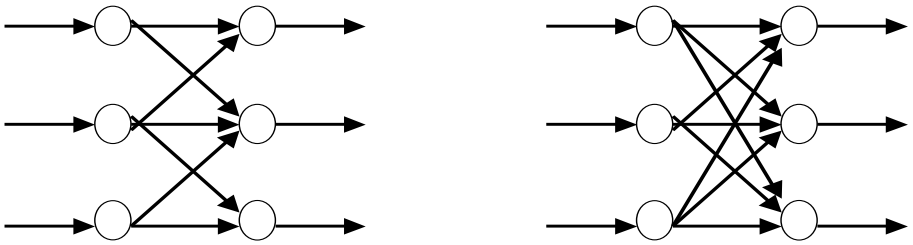
Le modèle de connexions définit la manière dont sont inter-connectés les neurones. En se basant sur la structure à couches, on distingue différents types de connexions : les connexions inter-couches (entre neurones de couches adjacentes), les connexions supra-couche (lorsque les couches ne sont pas adjacentes), les connexions intra-couche (entre neurones d'une même couche) et l'auto-connexion (un neurone avec lui même).



(a) Connexions directes (b) Connexions récurrentes

FIG. *1.8: Connexions directes et récurrentes.*

De manière générale, le sens de transfert de l'information dans un réseau est défini par la nature des connexions : directes ou récurrentes.
Les connexions directes sont celles qui sont dirigées d'une couche d'indice inférieur vers une couche d'indice supérieur (voir la figure 1.8 (a)).
Les connexions sont dites récurrentes lorsque les sorties de neurones d'une couche sont connectées aux entrées d'une couche d'indice inférieur (voir la figure 1.8 (b)).



(a) Connexions partielles (b) Connexions totales

FIG. *1.9: Connexions partielles et totales.*

Par ailleurs, entre deux couches, les connexions peuvent être partielles ou totales. L'utilisation de connexions partielles permet de regrouper certaines zones du réseau pour effectuer une fonction spécifique (voir la figure 1.9).

Apprentissage

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. C'est une phase de développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement.
Dans le cas des réseaux de neurones artificiels, on ajoute souvent à la description du modèle, l'algorithme d'apprentissage. Un modèle sans apprentissage présente en effet peu d'intérêt. Dans la majorité des algorithmes actuels, les variables modifiées pendant l'apprentissage sont les poids des connexions. L'apprentissage consiste à modifier les poids du réseau dans l'optique d'accorder la réponse du réseau aux exemples et à l'expérience. Il est souvent impossible de décider *a priori* des valeurs des poids des connexions d'un réseau pour une application donnée. A l'issu de l'apprentissage, les poids sont fixés : c'est alors la phase d'utilisation. Dans les modèles de réseaux de neurones conçus pour effectuer un apprentissage, on peut toujours distinguer une phase d'apprentissage (qui remet à jour le comportement) et une phase d'utilisation. Cette technique permet au réseau de conserver un comportement adapté malgré les fluctuations dans les données d'entrées.

Au niveau des algorithmes d'apprentissage, deux grandes classes ont été définies selon que l'apprentissage est dit supervisé ou non supervisé (Anouar, 1996). Cette distinction repose sur la forme des exemples d'apprentissage. Dans le cas de l'apprentissage supervisé, les exemples sont des couples (Entrée, Sortie associée) alors que l'on ne dispose que des valeurs (Entrée) pour l'apprentissage non supervisé.

Dans notre travail, nous nous sommes intéressés à un modèle particulier de réseaux de neurones artificiels qui est celui des cartes auto-organisatrices (SOM : Self Organizing Maps) (Kohonen, 1995). Ce modèle effectue une classification des données en utilisant un apprentissage non supervisé.

1.5.3 Cartes auto-organisatrices de Kohonen

Les cartes auto-organisatrices sont connues depuis longtemps (1977), mais ce n'est que très récemment (1990) que des applications les utilisent : carte phonétique, diagnostic de pannes, compression d'images, robotique, etc. Les cartes auto-organisatrices sont parmi les modèles connexionnistes les plus utilisés pour la classification et la visualisation des données (Kohonen, 1995). Elles permettent de projeter des données de dimension quelconque dans un espace discret de faible dimension. La carte SOM est composée de deux couches de neurones : une couche d'entrée et une couche compétitive de sortie (voir la figure 1.10 (c)). La couche compétitive est celle qui s'occupe du traitement des données d'entrée. Sa structure topologique est prédéfinie, la plus utilisée est celle sous forme d'une grille rectangulaire (voir la figure 1.10 (a)). L'algorithme d'apprentissage de SOM est essentiellement effectué suivant trois phases :

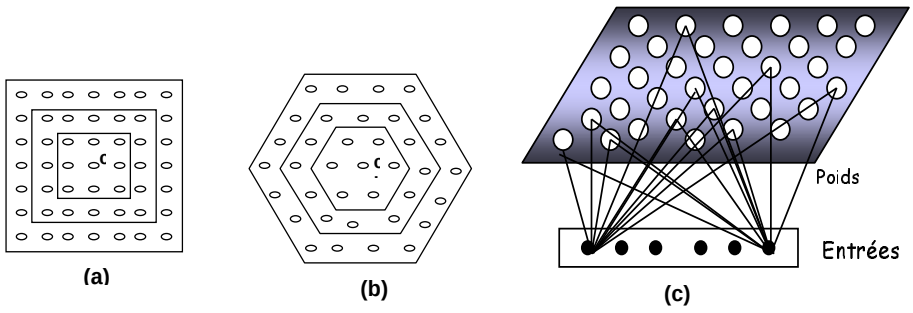


FIG. *1.10: (a),(b) Exemples de voisinages. (c) Architecture à deux couches d'une carte topologique de dimension 2.*

- La phase d'initialisation où des valeurs aléatoires sont affectées aux poids des neurones de sortie de la carte ;
- La phase de compétition dans laquelle pour toute entrée ξ, un neurone c_s est sélectionné

de la manière suivante :

$$c_s = \arg\min_{c_r \in A} dist(\xi, w_r) \tag{1.1}$$

où A est l'ensemble des neurones de sortie et dist(,) est la distance euclidienne ;

– La phase d'adaptation où les poids de chaque neurone de sortie c_r sont mis à jour de la manière suivante :

$$\Delta w_r = \epsilon(t) h_{rs}[\xi - w_r] \tag{1.2}$$

où $\epsilon(t)$ est le pas d'apprentissage et h_{rs} est une fonction de voisinage.

La fonction de voisinage la plus souvent utilisée est la fonction gaussienne (voir la figure 1.11). Formellement, la fonction de voisinage gaussienne entre deux neurones c_r et c_s est définie par :

$$h_{rs} = \exp(\frac{-d_1(r,s)^2}{2\sigma^2}) \tag{1.3}$$

où : σ est l'écart type et $d_1(,)$ est la distance de Manhattan (Anouar, 1996).

Cette fonction détermine le degré d'influence du neurone c_s sur le neurone c_r. La valeur de cette fonction dépend de la distance entre les deux neurones sur la carte. L'utilisation de cette fonction introduit pour chaque neurone un voisinage global. La taille réelle de ce voisinage est limitée par l'écart type σ de la gaussienne. L'influence du neurone c_s sur les neurones qui se trouvent au delà de cette étendue est négligeable.

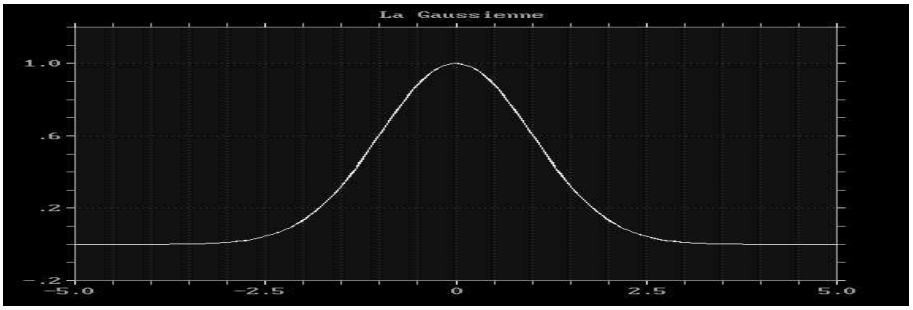


FIG. *1.11: Fonction de voisinage de type gaussien.*

1.5.4 Pourquoi choisir les cartes auto-organisatrices ?

Les cartes auto-organisatrices sont restées de nombreuses années ignorées malgré leur grand intérêt. Ce réseau de neurones a ensuite largement montré son efficacité. De nombreux raffinements et différentes variantes ont été élaborées.

L'intérêt des cartes de Kohonen est à la fois théorique et pratique, parce qu'elles introduisent la notion d'interaction entre les neurones et qu'elles utilisent des algorithmes relativement simples à mettre en œuvre.

Les neurones d'une carte auto-organisatrice sont disposés sur une grille régulière multi-dimensionnelle. La plupart des applications se limitent volontairement à des grilles ne dépassant pas trois dimensions. Au delà, les données deviennent difficilement interprétables.
La notion de grille de neurones est un artifice théorique qui permet d'introduire des interactions entre les neurones à travers un concept fondamental pour ce type de réseaux neuronaux : la notion de *voisinage*. Un neurone actif va ainsi pouvoir influencer les neurones les plus proches dans la grille. C'est une qualité essentielle des cartes, puisque l'apprentissage est à la fois rapide et stable.
L'originalité de la carte auto-organisatrice réside dans l'emploi d'une fonction de voisinage. Ce principe est tellement intéressant qu'il a été repris dans le développement de nouveaux réseaux. Leurs propriétés principales sont leurs interprétations géométriques.
Le vecteur de poids synaptiques associé à chaque neurone représente la position du neurone dans un espace. Cet espace est celui des entrées. D'un point de vue pratique, l'initialisation des poids se fait aléatoirement dans un espace contenu dans l'espace à apprendre. Pendant l'apprentissage, les neurones vont se déplacer dans l'espace d'entrée en déformant et en étirant la grille. Lorsque suffisamment de prototypes de l'espace d'entrée ont été présentés, les neurones de la grille sont, après apprentissage, répartis de façon plus ou moins régulière dans l'espace d'entrée, réalisant ainsi une discrétisation naturelle.
Chaque neurone représente alors un centre d'activation. Les sorties de l'architecture SOM définissent les régions d'influence comme étant l'ensemble des points les plus proches des poids au sens de la distance utilisée. L'ensemble de ces régions réalise une partition de l'espace qui dépend de la distance utilisée. La carte s'adapte dynamiquement à l'espace des entrées et en détermine le meilleur pavage.
En plus de cette propriété de quantification ou de discrétisation, les cartes SOM permettent de préserver les propriétés topologiques de l'espace d'entrée. Autrement dit, deux neurones voisins dans la grille sont représentatifs de deux régions voisines dans l'espace d'entrée.
Une interprétation statistique conclue qu'après un apprentissage correct d'une distribution de points, la distribution des vecteurs de référence reflète celle de l'espace d'entrée.
Il est à remarquer que dans le principe de fonctionnement d'une carte SOM, en plus d'une phase d'apprentissage, on retrouve deux grands principes récurrents dans le domaines des réseaux neuronaux : la compétition et la coopération.

- La compétition lors du choix du neurone vainqueur (voir l'équation 1.1) : il s'agit de distinguer parmi tous les neurones, le plus apte à répondre au stimulus de l'entrée. Ceci est traduit dans un sens géométrique par le plus proche neurone du vecteur d'entrée.
- La coopération pour le partage des connaissances à l'aide de la fonction de voisinage (voir l'équation 1.2) : lorsqu'un neurone est choisi comme centre d'activation de la carte, il partage ses connaissances avec ses voisins les plus proches en les forçant à s'adapter, c'est à dire en se rapprochant au sens géométrique des cartes auto-organisatrices. Ceci rend ces réseaux neuronaux très intéressants dans plusieurs applications.

1.5.5 Utilisation des cartes auto-organisatrices pour notre classe de problèmes

Les cartes auto-organisatrices peuvent être utilisées pour traiter de manière efficace de gros volumes de données sans utiliser des connaissances *a priori* du domaine. Donc les contraintes temps réel peuvent être satisfaites.
Les cartes auto-organisatrices classiques traitent des données statiques et ne considèrent pas l'aspect séquentiel des données. Plusieurs variantes de ces cartes ont été proposées pour traiter les séquences temporelles mais ces variantes ne sont pas capables d'acquérir de nouvelles connaissances tout en préservant les anciennes.
D'autres variantes des cartes auto-organisatrices, qui sont évolutives, ont aussi été proposées dans la littérature. Celles-ci peuvent acquérir de nouvelles connaissances et s'adapter aux nouvelles données, mais il n'y a aucune garantie concernant la préservation des anciennes connaissances déjà acquises.

Nous avons présenté les cartes auto-organisatrices et les motivations de leur utilisation pour notre classe de problèmes. Mais, comme nous l'avons mentionné précédemment, ces cartes doivent vérifier certaines propriétés. En effet, notre système doit être capable de traiter les séquences temporelles. Il doit aussi apprendre de façon permanente, ce qui pose le dilemme plasticité/stabilité : il faut que le même réseau apprenne de nouvelles entrées sans oublier les anciennes. Pour ceci, nous nous sommes intéressés à la théorie de résonance adaptative « Adaptive Resonance Theory : ART » (Carpenter et Grossberg, 1987).

1.6 La théorie de résonance adaptative

Pour réaliser un système qui permette d'acquérir en continu de nouvelles connaissances (plasticité) tout en continuant à mémoriser les anciennes (stabilité), Carpenter et Grossberg (Carpenter et Grossberg, 1987) ont proposé une théorie dite de la résonance adaptative « ART ».
Les modèles construits en utilisant cette théorie sont parmi les rares modèles qui peuvent apprendre dans un environnement variant continûment. Ces modèles protègent les connaissances acquises en rendant l'apprentissage conditionnel. Pour ce faire, ils distinguent la mémoire à long terme, contenue dans les connexions et modifiée par apprentissage, de la mémoire à court terme, mémoire instable contenue dans les neurones.
Si une entrée est reconnue, la mémoire à long terme correspondante devrait être modifiée pour se rapprocher de la forme d'entrée (plasticité). Dans un premier temps, seule la mémoire à court terme du réseau est modifiée sous l'influence de la mémoire à long terme. L'apprentissage n'aura lieu que si la mémoire à court terme est suffisamment proche de l'entrée ; dans ce cas, la mémoire à court terme deviendra une mémoire à long terme ; un processeur spécialisé (MO : module d'orientation) évalue la distance par rapport à un seuil d'attention qui est un paramètre de contrôle de l'évolution du réseau (stabilité).

Par conséquent, la phase d'attraction est ici une boucle, comprenant une phase dite de résonance : la couche de reconnaissance fournit une réponse, cette réponse est comparée avec l'entrée, puis le résultat de la comparaison est renvoyé au processeur MO ; tant que la réponse n'est pas assez proche de l'entrée, de nouvelles réponses sont demandées à la couche de reconnaissance. Si aucun neurone ne fournit une bonne réponse, un nouveau neurone est alloué.
L'architecture de ART (voir la figure 1.12) présente donc des connexions montantes et des connexions descendantes. Ces connexions lient les deux couches de comparaison et de reconnaissance. Les deux couches sont aussi connectées au processeur MO qui contrôle la résonance. Le fonctionnement des modèles ART est caractérisé par la boucle de résonance précédant la phase d'apprentissage. Un mécanisme de contrôle est aussi associé à chacune des deux couches.

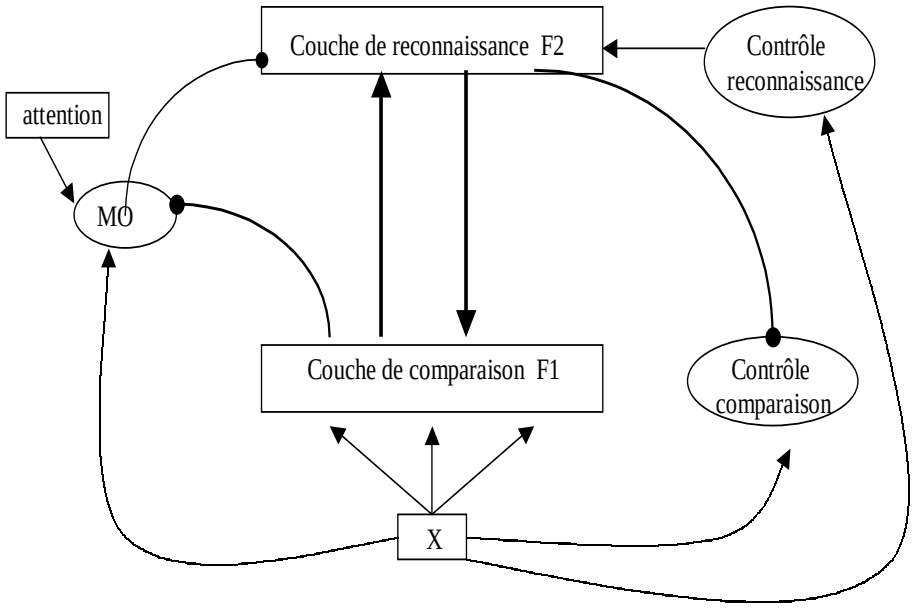


FIG. *1.12: Architecture de ART*

La théorie de résonance adaptative a donné naissance à plusieurs modèles comme : ART1 (Grossberg, 1987), ART2 (Carpenter et Grossberg, 1988), Fuzzy ART (Carpenter *et al.*, 1991), Artmap (Carpenter, 1997), Simplified ART (Baraldi et Alpaydin, 1998), etc.
Le modèle ART1 traite uniquement les entrées binaires, ce qui est très restrictif. Toutefois, d'autres modèles comme ART2, Fuzzy ART et Simplified ART remédient à ce problème en traitant des données réelles.
Nous avons introduit le paradigme de la théorie de résonance adaptative dans une carte auto-organisatrice pour la doter des propriétés de plasticité et de stabilité.

1.7 Vers l'hybridation du raisonnement à partir de cas et des cartes auto-organisatrices

Pour réaliser un système qui prenne en compte les contraintes de notre application, nous avons pensé à combiner le raisonnement à partir de cas avec des cartes auto-organisatrices. Ceci a été effectué dans le but de tirer parti des avantages de chaque technique. Les cartes auto-organisatrices pour leur efficacité dans le traitement de gros volumes de données en un temps raisonnable et leur capacité de tenir compte de la présence du bruit dans les données ; le RàPC pour son incrémentalité en temps réel (un cas ajouté à la base de cas peut tout de suite après être utilisé pour résoudre les nouveaux problèmes) et pour sa capacité à fournir des solutions précises dans certaines situations et à justifier ces solutions.

1.8 Plan de lecture

Le document est organisé en trois parties principales :

- Dans la première partie, nous présentons certaines approches existantes pour le traitement de séquences utilisant le raisonnement à partir de cas ainsi que les critères d'évaluation d'une base de cas et différentes approches existantes sur la maintenance de la base de cas. Nous décrivons ensuite, le système de RàPC pour le classement (ou la prédiction) de séquences « CASEP », nous comparons notre approche aux approches existantes et nous décrivons les tests effectués pour valider le système CASEP.

- Dans la deuxième partie, nous décrivons les approches existantes pour le traitement des séquences temporelles en utilisant les réseaux de neurones, des cartes topologiques évolutives et certains réseaux du modèle ART. Nous présentons ensuite, de nouveaux modèles de cartes auto-organisatrices que nous avons proposés ainsi que les résultats de leur validation.

- Dans la troisième partie, nous décrivons les systèmes hybrides neuro-symboliques en général, puis nous donnons quelques exemples de systèmes hybrides qui combinent le RàPC avec les réseaux de neurones. Une description générale du système hybride « CASEP2 » que nous avons proposé est donnée ainsi que les expérimentations effectuées pour sa validation.

Première partie

Le raisonnement à partir de cas pour notre classe de problème

Chapitre 2

Approches de RàPC : État de l'art

Dans ce chapitre, nous décrivons certains systèmes de RàPC qui traitent les séquences ainsi que différentes stratégies de la maintenance de la base de cas qui permet de faire face au gros volume de données et au bruit contenu dans ces données.

2.1 Raisonnement à partir de cas et traitement de séquences

Certains systèmes existants, utilisant le RàPC, ont abordé le problème de traitement de séquences ou des historiques. Nous pouvons classer ces systèmes suivant la représentation du cas utilisée, suivant les mesures de similarité utilisées dans la phase de recherche et suivant leur phase de réutilisation.

Concernant la représentation de cas :

- Un premier classement consiste à distinguer les systèmes où les cas sont représentés par une succession d'instants ou d'événements instantanés (approche à base de points ou d'instants) des systèmes qui représentent le cas par des relations entre des intervalles temporels.
- Un deuxième classement consiste à distinguer les systèmes qui utilisent la séquence entière de ceux qui utilisent une expérience précise dans la séquence pour la représentation d'un cas source.
- Un troisième classement distingue les cas décomposables des cas monolithiques. Un cas peut être décomposé en morceaux où chaque morceau est indexé séparément. La décomposition permet de retrouver des similarités entre le cas cible et des parties de cas sources différents. Un cas peut aussi être non décomposable et dans cette situation, seuls les cas entiers sont remémorés.

Les systèmes de RàPC peuvent aussi être différenciés par les mesures de similarité utilisées dans la phase de recherche. Ces mesures dépendent de la représentation de cas utilisée. En plus du fait que ces mesures dépendent du domaine d'application, les types de mesures utilisés quand

les cas sont de même longueur sont différents de ceux utilisés quand les cas sont de longueurs différentes. Dans cette dernière situation, les mesures les plus utilisées sont des développements de l'algorithme standard de calcul de distance d'édition par la programmation dynamique (Bellman, 1957) qui effectue un alignement des séquences pour assurer une similarité maximale (Boyer *et al.*, 1987).

Concernant la phase de réutilisation, nous pouvons distinguer les systèmes qui prennent en compte les différents traitements effectués dans une même séquence de ceux qui considèrent chaque nouveau cas dans une même séquence comme indépendant du précédent.

L'approche de recommandation BROADWAY est la première approche ayant utilisé la représentation de cas sous forme d'expériences précises dans une séquence. Parmi les applications de cette approche, nous pouvons citer :

- Le système BroadwayV1 (Jaczynski et Trousse, 1998) qui est un assistant de navigation sur le Web. Ce système est accessible par un groupe d'utilisateurs et permet ainsi une collaboration indirecte en réutilisant les cas issus des navigations du groupe. Dans ce système un cas est une expérience précise extraite des navigations antérieures avec une évaluation des situations comportementales. Chaque cas est monolithique et constitué d'une succession d'instants.
- Le système BeCBKB (Kanawati *et al.*, 1999) dont le but est le raffinement des requêtes dans le contexte de moteur de méta-recherche CBKB en rajoutant les services de prédiction fournis par l'approche BROADWAY. Dans ce système, un cas représente une expérience précise dans une session de recherche. La partie problème est constituée d'une partie instantanée composée du contexte de la session à partir de laquelle le cas est extrait et d'une partie comportementale contenant les p dernières requêtes observées, l'ensemble des documents retenus ou exclus et les dernières configurations des requêtes qui ont mené à l'échec. La partie solution est composée des recommandations positives et négatives. La représentation monolithique d'un cas sous forme d'une succession d'instants est aussi utilisée dans ce système.

Dans les deux systèmes cités, la fonction de similarité utilisée est une agrégation de similarités entre les composants de la séquence tout en tenant compte de leur ordre dans la séquence.

Le système COBRA (Malek et Kanawati, 2001) effectue les prédictions des actions des utilisateurs d'un site Web. L'approche modélise les comportements des utilisateurs d'un site Web par un ensemble de cas. La partie problème d'un cas représente une expérience de navigation. La partie solution est composée d'un ensemble d'actions qui peuvent expliquer la transition (déplacement d'une page à une autre) qui suit l'expérience de navigation décrite dans la partie problème du cas. Un cas dans ce système est monolithique et sa représentation est à base d'instants. L'ordre

des composants de la séquence est pris en compte dans le calcul de la fonction de similarité qui est une agrégation de similarités entre ces composants.

Le système Creek (Jaere *et al.*, 2002) permet de prédire les évènements non désirés dans le forage des puits de pétrole. L'approche utilisée dans ce système est celle à base d'intervalles qui fournit une représentation riche du cas. La description d'un cas contient les différentes relations qui lient les intervalles temporels dans lesquels les événements se sont produits. Les relations entre ces intervalles sont exprimées par la logique temporelle à base d'intervalles proposée par Allen (Allen, 1983). Un cas est représenté par un réseau temporel (Jaere *et al.*, 2002) qui contient toutes ces relations. Les cas du système sont monolithiques et représentent des séquences entières. Trois mesures de similarité, qui ne prennent pas en compte l'aspect temporel des cas sont d'abord utilisées pour remémorer des cas. Ensuite, une mesure de similarité temporelle supplémentaire est utilisé. Celle-ci permet de comparer tous les chemins temporels contenus dans le cas. Comme plusieurs chemins temporels peuvent être formés par combinaison de relations primitives, l'espace de recherche est réduit selon le but de la recherche.

REBECAS (Rougegrez-Loriette, 1994) est un système qui effectue la prédiction de l'évolution des incendies de forêts. Ce système permet de prendre en compte les évolutions de l'environnement sous forme d'historiques d'événements pour essayer de donner une prévision. Un cas est constitué de l'ensemble des données d'observation disponibles pour le déroulement complet d'un feu et il est décomposé suivant plusieurs directions de propagation de celui-ci. Un cas dans ce système est monolithique représenté par une séquence entière constituée d'une succession d'instants. Comme les cas dans ce système sont de longueurs différentes, une mesure de similarité basée sur un algorithme d'alignement de séquences est utilisée.

Le système SINS (Ram et Santamaria, 1997) réalisé pour la commande d'un robot dans un environnement réel utilise aussi des cas monolithiques représentant des séquences entières sous forme d'instants. Le système prend en compte les historiques de l'évolution des paramètres décrivant une situation. Deux types de variables sont identifiés : les variables de mesures qui décrivent l'environnement et les variables qui représentent les paramètres des schémas de réaction. Un cas représente les évolutions des variables issues d'expériences passées pour une durée fixée. La similarité entre les cas est basée sur le calcul de la moyenne quadratique des différences entre le vecteur représentant le cas dans une fenêtre temporelle et celui qui représente l'environnement.

Radix (Corvaisier *et al.*, 1997) est système d'aide à la recherche d'information sur le Web. Un cas représente une session de navigation composée d'un ensemble ordonné de tentatives unitaires de recherche (TU). Chaque TU se décompose à son tour en une séquence d'unités de recherche reliant deux pages Web par une transition. Comme dans les deux systèmes précédents, un cas est monolithique représentant la totalité de la séquence sous forme d'instants. Dans ce système, deux mesures de similarité sont utilisées dans la phase de recherche. La première mesure basée

sur une similarité de surface permet de faire un filtrage des sessions en fonction du contexte de la session en cours. La deuxième, basée sur les connaissances de la session, permet de choisir parmi les sessions filtrées la plus facile à utiliser dans le cadre de l'adaptation. C'est une agrégation de plusieurs mesures de similarité dont une mesure qui combine des mesures de représentativité et de dispersion dans la session.

Le système PADIM (Fuchs *et al.*, 1995) aide un opérateur dans la supervision d'un procédé complexe en lui proposant les descriptions visuelles les plus adaptées. Un cas source est indexé par une situation composée d'une partie instantanée donnant l'état du système, le profil de l'utilisateur et le but de la supervision et d'une partie comportementale regroupant les événements pertinents issus de l'évolution du procédé et des actions de l'opérateur. La solution du cas source représente un environnement de supervision. Un cas dans ce système est monolithique et représente toute la séquence qui constitue une succession d'instants. L'espace de recherche de cas est d'abord réduit par un filtrage sur la partie instantanée, puis une sélection est effectuée en prenant en compte la partie comportementale. La mesure de similarité pour comparer deux séquences suppose que les événements de la séquence du cas cible se retrouvent tous dans la séquence d'un cas source et dans le même ordre. Cette mesure prend en compte la représentativité et la dispersion relative des deux séquences comparées.

Le système CELIA (Redmond, 1990) implémente un modèle d'acquisition progressive d'expériences de résolution de problèmes afin de compléter un modèle initialement incomplet. Il simule la mémoire et les capacités de raisonnement d'un apprenant en diagnostic. Les connaissances du système évoluent en écoutant les leçons données par un enseignant. Un cas représente le raisonnement de l'enseignant durant la leçon. Chaque cas est structuré selon différents buts. Chaque action est considérée comme la réalisation d'un but de l'enseignant et la prédiction porte sur les actions liées au prochain but. Un cas dans ce système est structuré en fragments représentant un raisonnement de l'enseignant. Les fragments sont indexés individuellement et contiennent le contexte dans lequel ils ont été utilisés, ce qui permet d'y accéder directement. Le cas est donc décomposable puisque des fragments provenant de cas sources différents peuvent être remémorés, puis combinés pour fournir une solution au cas cible.

Concernant la phase d'adaptation des systèmes décrits ci-dessus, les différents traitements effectués dans la même séquence ne sont pas réutilisés dans la phase d'adaptation. Le traitement de deux cas consécutifs d'une même séquence est effectué de la même manière que celui de deux cas de deux séquences différentes.

2.1.1 Discussion

La représentation d'un cas sous forme d'une séquence d'événements instantanés tient compte de manière assez simple des informations temporelles contenues dans le cas. Cette représentation a été utilisée avec succès dans plusieurs domaines. La représentation de cas sous forme de relations entre des intervalles temporels est une nouvelle approche qui n'a pas encore été beaucoup utilisée. Cette approche est, certes, plus complexe que la première, mais beaucoup plus riche et permet de représenter plus d'informations dans un cas. La complexité de la structure du cas peut représenter certains inconvénients, surtout face à un gros volume de données et à des contraintes de temps comme c'est le cas dans notre classe de problème. Cependant, nous pensons que cette représentation est intéressante pour sa richesse et la possibilité de son utilisation dans des domaine où la représentation d'un cas sous forme d'une succession de points peut ne pas donner de bons résultats.
La représentation d'un cas sous forme d'une séquence entière évite un stockage supplémentaire de connaissances plus précises (moins d'espace mémoire), mais son inconvénient est que dans la plupart des systèmes utilisant cette représentation, aucune leçon n'est tirée des différents raisonnements effectués durant l'évolution de la séquence comme dans le système REBECAS où la seule opération effectuée pour mettre à jour les connaissances du système est l'ajout de la séquence complète à la base de cas. Le fait de tenir compte des différents raisonnements effectués durant l'évolution de la séquence comme dans SINS (Ram et Santamaria, 1997) et Broadway (Jaczynski, 1998) permet d'améliorer les résultats du système et d'éviter les échecs précédents.
La représentation d'un cas sous forme d'une expérience dans une séquence peut nécessiter plus d'espace mémoire quand les séquences entières et les cas sont gardés dans la mémoire. L'avantage de cette représentation est qu'à chaque raisonnement effectué, les connaissances utiles sont acquises par le système. Ceci permet de réduire le temps de remémoration de cas et aussi d'améliorer les résultats fournis par le système.
Quant à la représentation de cas par morceaux, elle permet de retrouver des similarités entre des parties de cas différents et ainsi peut permettre la généralisation des enseignements, mais ne peut être utilisée dans notre application.
Les différents systèmes cités se sont peu intéressés aux problèmes liés à la maintenance du système de RàPC et principalement au contrôle de la taille et du contenu de la mémoire du système. Dans le paragraphe suivant (§2.1.2), nous allons aborder le problème de la maintenance des systèmes de raisonnement à partir de cas.

2.1.2 Vers une maintenance du système de RàPC

Nous avons conçu un système de RàPC pour le classement (ou la prédiction) de séquences qui contient dans sa mémoire des cas (expériences précises) et des séquences entières. Donc la mémoire de notre système contient une base de cas et une base de séquences. Nous avons effectué ce choix, car comme nous l'avons souligné précédemment, l'utilisation de la base de cas toute seule risque de détériorer la qualité des résultats fournis par le système et l'utilisation de la base

de séquences toute seule risque de rendre la phase de recherche coûteuse en temps puisque les cas sources doivent être extraits à partir de séquences à chaque cycle de RàPC.
Durant la phase de recherche, nous avons favorisé l'utilisation de la base de cas et nous avons utilisé la base de séquences pour résoudre les cas cibles non résolus par le système en utilisant la base de cas. Nous disposons d'un grand nombre de séquences dans la base de séquences. De plus, de chaque séquence un grand nombre de cas peut être extrait (dans notre application, des milliers de navigations (séquences) sont enregistrées chaque jour), donc la base de cas aussi risque d'être volumineuse.
Pour faire face à ces problèmes, la maintenance du système de RàPC est nécessaire. Cette maintenance consiste à développer des techniques afin de contrôler et réagir face aux changements des différentes sources de connaissances du système de RàPC. Ces sources représentent des connaissances pertinentes du système (Leake et Wilson, 2002). Les plus importantes sont :

- *La source de vocabulaire :* contient toutes les informations sur les définitions et les structures utilisées dans le système de RàPC.
- *La source des mesures de similarité :* contient les mesures nécessaires pour la recherche des cas.
- *La source d'adaptation de solutions :* contient les connaissances pour transformer les solutions des cas remémorées afin de résoudre de nouveaux problèmes.
- *La source de la base de cas :* représente le contenu et l'organisation de la base de cas.

Chacune de ces sources de connaissances peut être maintenue séparément, mais elles sont interdépendantes : les connaissances disponibles dans l'une peuvent remplacer les connaissances manquantes dans l'autre (Leake et Wilson, 2002). Par exemple, le même effet général sur la précision du système peut être obtenu par réorganisation de la base de cas, ce qui est considéré comme une maintenance de la base de cas, ou par l'ajustement des mesures de similarité, ce qui est considéré comme maintenance des mesures de similarité.
La maintenance devient nécessaire pour des systèmes qui sont conçus pour fonctionner sur des longues périodes et/ou qui seront amenés à traiter de grands volumes de données et de cas. L'amélioration visée par la maintenance peut être mesurée par deux critères :

- **La performance** du système mesurée par le temps de réponse qui lui est nécessaire pour proposer une solution à un cas cible. Dans certaines applications (i.e. assistants de recherche, prédiction de comportement, etc.), ce temps doit être le plus court possible.
- **La compétence** mesurée par le nombre de problèmes différents pour lesquels le système apporte une bonne solution.

La plupart des travaux concernant la maintenance des systèmes de RàPC se sont intéressés à la maintenance de la base de cas (MBC).

2.2 Maintenance de la base de cas

La MBC est définie comme étant le processus de raffinement de la base de cas dans les systèmes de RàPC.
Elle met en œuvre des politiques pour réviser l'organisation et/ou le contenu de la base de cas afin d'améliorer le raisonnement futur (Leake et Wilson, 1998).
L'intérêt de la maintenance de la base de cas peut être justifié par le fait qu'aucune maintenance ne peut être effectuée sans consulter la base de cas. De plus, les connaissances d'un système de RàPC sont liées aux cas puisque ceux-ci sont affectés par tout changement dans les sources de connaissances (par exemple, en ajoutant des connaissances aux mesures de similarité, les cas seront applicables à de nouvelles situations). La base de cas étant la source de connaissances la plus sensible aux changements dans le système de RàPC, sa consultation est la plus appropriée pour déclencher les opérations de maintenance. Comme le système de RàPC se comporte en fonction du contenu et de la taille de sa base de cas, celle-ci permet de détecter certains problèmes comme par exemple des tendances ou des régions importantes non traitées, et donc de faire une maintenance préventive.
La base de cas a donc un rôle très important dans la maintenance des systèmes de RàPC.
Dans les applications réelles où l'environnement est dynamique, les types des problèmes cibles peuvent changer dans le temps, ce qui limite la performance de la base de cas existante. L'ajout de nouveaux cas peut mener à des inconsistances ou à des redondances dans la base de cas. Les stratégies de maintenance de la base de cas doivent être conçues pour contribuer à reconnaître et à résoudre ces problèmes en supprimant, en ajoutant ou en modifiant des cas.
Nous nous sommes intéressés dans notre travail à effectuer la maintenance de notre système de RàPC tout en nous focalisant sur la maintenance de sa base de cas.

2.2.1 Définitions et analyse de la maintenance de la base de cas

D.B. Leake et D.C. Wilson (Leake et Wilson, 1998) ont présenté une première analyse du processus de maintenance de la base de cas (MBC). Cette analyse catégorise les approches de la MBC en terme de politiques qui déterminent quand et comment un système de RàPC exécute la MBC (voir la figure 2.1). Ces politiques sont décrites par :

- La manière de rassembler les données pertinentes pour la maintenance ;
- Le moment du déclenchement de la maintenance ;
- Les types d'opérations de maintenance possibles ;
- La manière d'exécuter les opérations sélectionnées.

L'intégration de la collecte de données, du déclenchement de la MBC et de son exécution peut s'effectuer en ligne pendant l'utilisation du système pour la tâche visée ou bien hors ligne en dehors du moment de son utilisation.
De plus le cadencement (la fréquence d'exécution) de ces trois opérations peut être périodique,

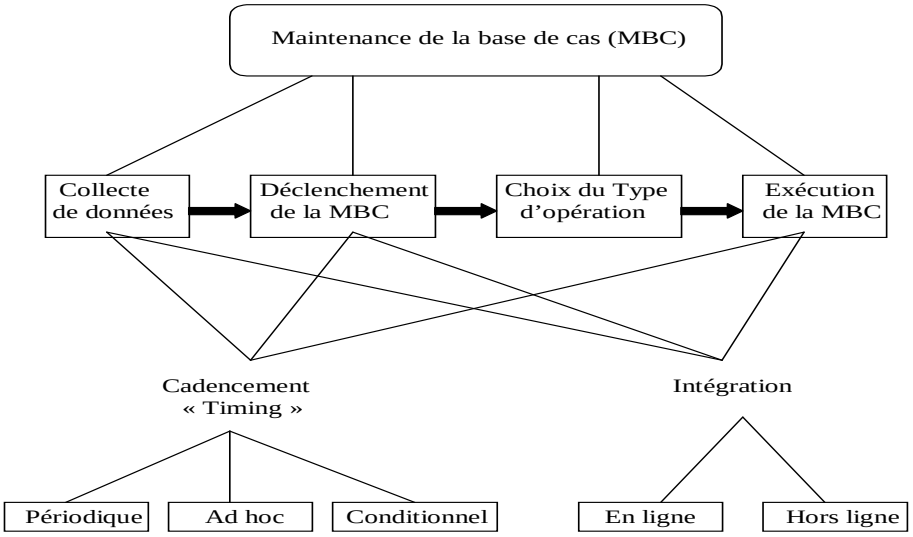


FIG. *2.1: Catégorisation de la maintenance de la base de cas selon le cadencement et le mode d'intégration des différentes phases.*

conditionnel (à la satisfaction de certaines contraintes) ou bien ad hoc.

T. Reinartz et al. (Reinartz *et al.*, 2000) ont défini deux nouvelles phases dans le cycle du RàPC (figure 2.2). Dans la représentation du cas, un troisième champ est ajouté : « information de qualité » qui contient toutes les données nécessaires pour effectuer la maintenance de la base de cas. Les phases ajoutées au cycle de RàPC sont : *la phase d'examen* et *la phase de restauration.*

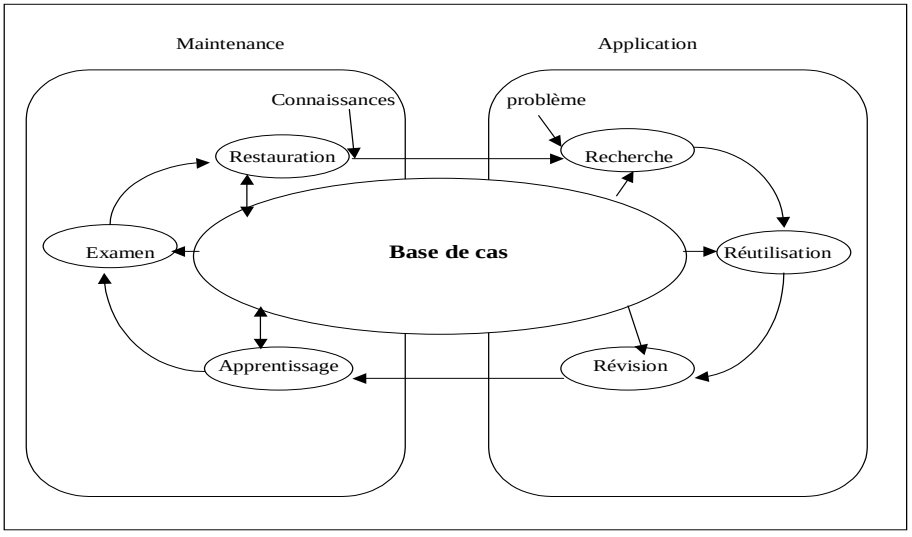


FIG. *2.2: Ajout de deux phases au cycle de RàPC*

– **La phase d'examen**
Cette phase permet d'effectuer la collecte de données et de déclencher la maintenance de la base de cas à chaque cycle de RàPC pendant le fonctionnement du système (en ligne). Elle permet aussi de proposer des types d'opérations de maintenance pour la phase de restauration.
Elle considère l'état courant de la base de cas et évalue sa qualité (les critères d'évaluation de la qualité de base de cas sont cités dans le paragraphe §2.2.2). Une mesure de qualité est définie pour permettre le calcul des valeurs qui évaluent l'état de la base de cas. Ces valeurs sont alors contrôlées et des indicateurs spécifiques mènent à l'initialisation de l'étape de restauration. Si la qualité de la base de cas est mauvaise, la phase d'examen suggère des changements spécifiques pour avoir la qualité désirée.

– **La phase de restauration**
Si les valeurs de qualité restent satisfaisantes, la base de cas reste inchangée et une nouvelle itération du cycle de maintenance est relancée.
Sinon, des méthodes de sélection sont utilisées dans cette phase pour choisir un opérateur de modification parmi ceux suggérés par la phase d'examen afin d'avoir le niveau de qualité requis. Cette phase utilise ensuite ces opérateurs pour changer le contenu de la base de cas. Quand un opérateur de modification est sélectionné (ajout de cas, suppression de cas, etc.), il peut être exécuté automatiquement ou avec l'aide de l'utilisateur. Après cette modification, la qualité de l'information est mise à jour pour refléter le changement.

Après la collecte de données, celles-ci sont analysées pour déclencher des opérations de maintenance. Cette analyse porte sur l'évaluation de la qualité de la base de cas. Plusieurs propriétés des cas et de la base de cas ont été proposées pour effectuer cette évaluation.

2.2.2 Évaluation de la qualité d'une base de cas

Pour effectuer la MBC, une évaluation de la qualité de la base de cas est nécessaire. Plusieurs critères qui permettent de juger cette qualité ont été proposés dans la littérature (Reinartz *et al.*, 2000), (Roth-Berghofer et Iglezakis, 2001), (Racine et Yang, 1996) et (Smyth et Keane, 1995b). Une base de cas est de bonne qualité si elle permet au système de RàPC de résoudre le plus de problèmes possibles de manière correcte en un temps raisonnable (sa durée dépend de l'application). Plus précisément, une base de cas peut être évaluée par les critères suivants :

1. **Consistance**
Deux sortes de consistances (Racine et Yang, 1996) sont identifiées :

– *consistance intra-cas :* une consistance intra-cas est vérifiée quand les valeurs affectées à différentes caractéristiques dans un seul cas satisfont toutes les contraintes liées à ces ca-

ractéristiques (comme par exemple, l'appartenance des valeurs au domaine de définition des caractéristiques).

- *consistance inter-cas :* un cas $cas_1 = ((P_1), (S_1))$ est dit consistant avec un ensemble E de cas s'il n'existe pas de cas $cas_2 = ((P_2), (S_2)) \in E$ qui peut résoudre un problème plus général ($P_1 \subset P_2$) de manière différente ($S_1 \neq S_2$).

2. **Redondance**

   La redondance d'un cas dans la base de cas (Racine et Yang, 1996) est définie par la relation de subsomption. Un cas cas_1 est redondant dans la base de cas s'il est subsumé par un autre cas cas_2 de la base de cas.

   Formellement, la relation de subsomption entre deux cas : $cas_1 = ((P_1), (S_1))$ et $cas_2 = ((P_2), (S_2))$ est définie par :

   cas_2 subsume cas_1 si $p_1 \subset p_2$ et la solution de cas_2 est plus précise que celle de cas_1 ($S_2 \subset S_1$).

3. **Recouvrement/Compétence**

   Le recouvrement d'un cas de la base de cas (Smyth et Keane, 1995b) représente l'ensemble des cas cibles que ce cas peut résoudre. La compétence de la base de cas représente le recouvrement des cas qu'elle contient. C'est le nombre et le type de problèmes que le système peut résoudre.

   Plusieurs modèles de compétence qui permettent de mesurer le recouvrement d'un cas (ou la compétence de la base de cas) ont été proposés dans la littérature (Smyth et McKenna, 2002a), (Smyth et McKenna, 2002b), (Smyth et McKenna, 1998) et (Smyth et McKenna, 1999), ces modèles sont décrits dans le paragraphe §2.2.3.

4. **Atteignabilité**

   L'atteignabilité d'un cas (Smyth et Keane, 1995b) représente l'ensemble de cas qui peuvent le résoudre (qui le recouvrent). Cette mesure est liée à celle du recouvrement. Les modèles de compétence de la base de cas décrits dans le paragraphe §2.2.3 dépendent des deux mesures (recouvrement et atteignabilité).

5. **Performance**

   Cette mesure représente le temps mis par le système de RàPC pour résoudre un problème. Ce temps comprend principalement le coût de recherche de cas similaires au cas cible dans la base de cas et le coût d'adaptation des solutions de ces cas pour fournir une solution au cas cible.

   - *Coût d'adaptation*

     Le coût d'adaptation est le temps nécessaire pour adapter les solutions d'un ou plusieurs

cas remémorés pour résoudre le cas cible. Certaines stratégies de maintenance de la base de cas (Leake et Wilson, 2000) consistent à supprimer ou à ajouter des cas à la base de cas de façon à réduire le coût d'adaptation. Différentes manières de mesurer ce coût sont décrites dans le paragraphe §2.2.3.

- *Coût de la recherche*
  Cette mesure représente le temps mis par le système pour remémorer un ou plusieurs cas (suivant l'application) similaires au cas cible. Cette mesure dépend de la structure de la base de cas, du nombre de cas dans la base de cas et de la représentation des cas. La plupart des stratégies d'ajout ou de suppression de cas pour la maintenance de la base de cas visent à réduire le coût de la recherche dans la base de cas sans trop détériorer sa compétence.

6. **Degré d'abstraction**
   Une base de cas peut contenir des cas concrets ou des cas généralisés. Le degré d'abstraction de la base de cas est le niveau de généralisation des cas qu'elle contient.

Les premiers efforts pour comprendre et mesurer la compétence de la base de cas se sont concentrés sur les propriétés statistiques de la base de cas comme la taille et la densité.
Le nombre de cas contenus dans la base de cas est certainement une mesure triviale de la compétence, mais elle n'est pas toujours un bon prédicteur de celle-ci. Ceci est dû au fait que différents cas peuvent avoir des contributions différentes à la compétence de la base de cas (Smyth et Keane, 1995b).
La densité d'un cas contribue aussi à la compétence. Formellement, la densité d'un cas c dans un groupe de cas G est donné par (Smyth et McKenna, 1998) :

$$densite(c, G) = \frac{\sum_{c' \in G - \{c\}} Similarite(c, c')}{|G| - 1} \tag{2.1}$$

La contribution individuelle d'un cas appartenant à un groupe dense (où les similarités entre les cas sont grandes) dans la résolution de problèmes est inférieure à celle d'un cas appartenant à un groupe non dense.

Le recouvrement d'un cas est l'une des mesures les plus utilisées pour évaluer la qualité de la base de cas. Un exemple de la modélisation de la compétence de la base de cas est décrit ci-dessous.

Exemple de modélisation de la compétence de la base de cas

Des modèles de compétence de la base de cas ont été proposés par B. Smyth. et E. Mckenna (Smyth et McKenna, 2002a), (Smyth et McKenna, 2002b), (Smyth et McKenna, 1998) et (Smyth

et McKenna, 1999), ces modèles sont basés sur deux hypothèses :

- Les cas de la base de cas correspondent à un échantillon représentatif des problèmes cibles[3].
- L'espace des problèmes est régulier, c.à.d que les problèmes similaires ont des solutions similaires[4].

Dans l'un des modèles de compétence proposé par B. Smyth et E. McKenna (Smyth et McKenna, 2002b), la taille de l'ensemble de recouvrement d'un cas est une mesure de sa compétence locale.
Formellement, pour tout cas c de la base de cas C,
Ensemble recouvrement (c)={ $c' \in C$: résout (c,c')}
Ensemble d'atteignabilité (c) ={ $c' \in C$: résout (c',c)}

Une autre mesure, appelée recouvrement relatif (RC), estime la contribution unique d'un cas à la compétence de la base de cas.
Formellement, si un cas c' est recouvert par n autres cas, alors chacun de ces n cas va recevoir une contribution de $\frac{1}{n}$ de c' à la mesure de leur recouvrement relatif.
$Recouvrement\ relatif(c) =$

$$\sum_{c' \in ensemble\_recouvrement(c)} \frac{1}{|ensemble\_atteignabilite(c')|} \tag{2.2}$$

Un groupe de compétence est une collection de cas qui ont une contribution collective à la compétence globale de la base de cas. Cette contribution est indépendante des autres collections de cas (pas de chevauchement de recouvrements). L'idée clé de la définition d'un groupe de compétence est celle du recouvrement partagé.
Formellement, soient c_1, c_2 dans C,

$recouvrement\ partagé(c_1, c_2) \iff$
{*Ensemble recouvrement* (c_1) $\cup$ *Ensemble d'atteignabilité* (c_1)}$\cap$
{*Ensemble recouvrement* (c_2) $\cup$ *Ensemble d'atteignabilité* (c_2) } $\neq \phi$

Un groupe de compétence est un ensemble maximal de cas qui ont des recouvrements partagés.
Comme les cas d'un groupe ne contribuent pas tous à la compétence dans ce modèle, le recouvrement du groupe de compétence dépend d'un sous-ensemble du groupe appelé « traces du groupes (*group footprint*) » obtenu par une stratégie de suppression de cas. Ce sous ensemble recouvre tous les cas du groupe.
Dans (Smyth et McKenna, 2002a), le recouvrement d'un groupe est mesuré en sommant les

[3]Pour ceci, les cas sources sont utilisés à la place des cas cibles pour définir le recouvrement d'un cas en pratique.
[4]Cette hypothèse est celle utilisée habituellement pour justifier l'utilisation du RàPC.

valeurs du recouvrement relatif de sa trace. Une fois que le recouvrement de chaque groupe est calculé, le recouvrement global de la base de cas est obtenu en sommant les valeurs des recouvrements des traces de groupe. Ceci est justifié par le fait que chaque groupe contribue de manière indépendante au recouvrement. Les expérimentations effectuées dans (Smyth et McKenna, 2002a) montrent qu'il y a une relation de proximité entre les courbes de la compétence prédite (recouvrement) et de la vraie compétence.
La compétence de la base de cas d'un système de RàPC évolue avec l'arrivée des données. Cette évolution a été étudiée par E. McKenna et B. Smyth (McKenna et Smyth, 1999).

Évolution de compétence de la base de cas

D'après Smyth et McKenna (McKenna et Smyth, 1999), quand un cas est ajouté à la base de cas, un des quatre événements suivants se produit :

- Un nouveau groupe de compétence est créé.
- La taille et le recouvrement d'un groupe existant augmentent.
- Un ensemble de groupes existants sont regroupés pour former un nouveau super groupe.
- Un groupe existant augmente en taille mais pas en recouvrement.

En général, l'occurrence de chacun de ces quatre événements dépend de l'état courant de la base de cas. Les auteurs de (McKenna et Smyth, 1999) ont trouvé qu'il y a une relation entre la probabilité que l'une des situations se produise et le niveau de maturité de la base de cas. En particulier, ils ont montré que durant sa construction, la base de cas passe par quatre phases de développement différentes appelées : enfance, adolescence, âge adulte et vieillesse (voir la figure 2.3), tel que dans chaque étape, l'un des événements cités précédemment domine.

La figure 2.3 montre un exemple de l'évolution d'une base de cas dans un domaine particulier (domaine des voyages). L'évolution de la base de cas dépend du système de RàPC et du domaine d'application. L'étude de cette évolution, montre que l'ajout de cas à la base de cas n'augmente pas automatiquement la compétence de la base de cas. A l'âge adulte, la compétence maximale de la base de cas est atteinte. L'ajout de cas à la base de cas après cette phase, ne fera qu'augmenter le temps de recherche dans la base de cas sans augmenter la compétence.

REMARQUE.
L'évolution de la compétence décrite par les auteurs de (McKenna et Smyth, 1999) concerne un domaine où les données ne sont pas dynamiques et où une compétence maximale peut être atteinte. Dans les domaines où les données changent avec le temps, la détection de l'une de ces phases devient plus compliquée. Avant la maturité de la base de cas (âge adulte) par rapport à

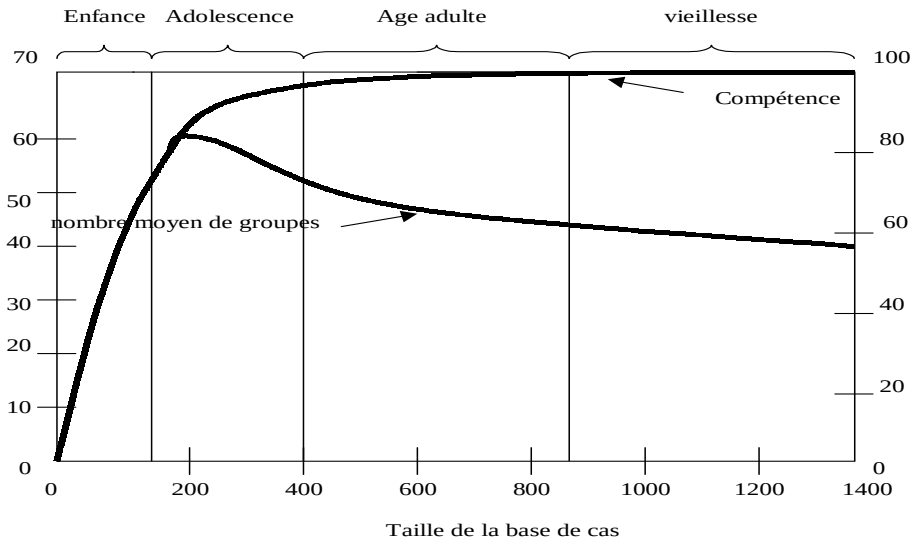


FIG. *2.3: Exemple de l'évolution de la base de cas dans le domaine de planification des voyages. Deux courbes sont représentées : l'une représente le nombre moyen de groupes formés et l'autre représente la compétence de la base de cas.*

certaines données, d'autres types de données peuvent surgir et un retour aux phases précédentes peut se produire.

2.2.3 Quelques stratégies de maintenance de la base de cas d'un système de RàPC

La plupart des approches proposées pour la MBC traitent le problème d'optimisation du contenu de la base de cas. Elles visent à réduire la taille de la base de cas sans détériorer les résultats fournis par le système.

La stratégie de suppression de cas consiste à supprimer des cas suivant une politique de suppression qui dépend des critères de la base de cas à optimiser (mesures de qualité de la base de cas). Smyth et Kean (Smyth et Keane, 1995b) ont introduit une approche pour préserver la compétence afin de guider leur politique de suppression de cas. Ceci est effectué à travers une classification de types de cas.

Une autre stratégie est celle de l'ajout de cas dans une nouvelle base de cas. Elle essaye d'améliorer l'efficacité en rendant disponible juste un sous ensemble de la base de cas pour le processus de remémoration (Yang et Zhu, 2001) et ce en construisant une nouvelle base de cas réduite.

Les méthodes utilisées pour chaque stratégie sont diverses. Elles peuvent considérer la compétence de chaque cas (Smyth et McKenna, 1998), ou l'effort d'adaptation que chaque cas nécessite (Leake et Wilson, 2000).

L'approche qui préserve la compétence a été aussi utilisée pour guider quelques politiques d'ajout de cas. Celles-ci consistent à choisir les cas de plus grande compétence à partir de la base de cas et à les rendre disponibles pour la phase de recherche. L'approche proposée dans (Smyth et

McKenna, 1999) est une politique d'ajout de cas qui construit une base de cas compacte avec une compétence élevée. Elle est basée sur la méthode CNN (Condensed Nearest Neighbor) et le recouvrement relatif (voir le paragraphe §2.2.3.1).
Une autre politique d'ajout de cas est proposée par Yang et Zhu (Yang et Zhu, 2001). Ceux-ci ont proposé un algorithme qui produit une base de cas dont le recouvrement n'est pas inférieur à 63% de celui d'une base de cas optimale.
De même que pour les approches qui préservent la compétence, des techniques qui préservent la performance d'un système de RàPC ont été étudiées. Celles-ci visent à minimiser le coût d'adaptation. Leake et Wilson (Leake et Wilson, 2000) ont proposé un algorithme qui préserve la performance appelé RP-CNN. L'approche est similaire à RC-CNN, mais basée sur la préservation de la performance de la base de cas initiale à travers une mesure appelée *performance relative*. Elle prend en compte le coût d'adaptation relatif de chaque cas.
Une autre stratégie de la maintenance de la base de cas, différente des précédentes, consiste à garder tous les cas de la base de cas volumineuse (aucune réduction de la base de cas n'est effectuée) et à la partitionner en groupes formant de petites bases de cas où la recherche de cas n'est pas coûteuse.
Yang et Wu (Yang et Wu, 2000) ont proposé une stratégie de maintenance basée sur deux idées principales :

- Partitionner la base de cas en groupes constitués de cas proches.
- Permettre à un utilisateur de rechercher les bases de cas distribuées en sélectionnant de manière incrémentale les attributs les plus pertinents.

Dans ce qui suit, nous allons décrire en détail les stratégies citées ci-dessus.

2.2.3.1 Stratégies d'optimisation de la base de cas

Les stratégies que nous allons citer dans ce paragraphe visent à réduire la taille de la base de cas afin d'atteindre certains objectifs comme des contraintes de temps de recherche, des limites dans la taille de la base de cas ou autres.

Stratégies de suppression de cas

Comme nous l'avons décrit précédemment, ces stratégies consistent à supprimer certains cas d'une base de cas volumineuse. Les cas supprimés sont ceux qui sont jugés les « moins importants » dans la base de cas selon certains critères qui seront cités ci-dessous.
Avec l'augmentation de la taille de la base de cas, celle-ci devient saturée et la performance du système de RàPC est réduite : c'est ce qui est appelé « problème d'utilité ».
La performance du système représente le temps mis par ce système pour trouver une solution à un cas cible : c'est principalement le temps de remémoration de cas et celui de l'adaptation.
Quand la taille de la base de cas augmente, le temps de recherche augmente et le temps d'adap-

tation diminue. Mais la diminution de ce dernier est moins importante que l'augmentation du temps de recherche, ce qui conduit à l'augmentation du temps de résolution de problèmes et donc à la diminution de la performance du système (Smyth et Cunningham, 1996).
Parmi les stratégies de suppression proposées dans la littérature, nous pouvons citer la politique de suppression aléatoire. Elle consiste à supprimer aléatoirement un cas de la base de cas dès que la taille de celle-ci dépasse une limite prédéfinie. Cette politique simple peut être aussi efficace que d'autres méthodes plus compliquées basées sur des principes décrits dans (Smyth, 2000). Parmi ces méthodes, nous trouvons celle qui utilise la mesure d'utilité de Minton (Smyth et Cunningham, 1996). Celle-ci consiste à choisir un cas à supprimer de la base de cas en se basant sur l'estimation de la performance de la base de cas.
Dans les systèmes de RàPC purs (sans connaissances *a priori* du domaine), l'application de ces politiques peut avoir des conséquences désastreuses sur sa compétence. En effet, les cas dans les systèmes de RàPC ne sont pas tous égaux. Quelques cas contribuent principalement à la compétence du système et d'autres contribuent peu à cette compétence. La suppression des premiers constitue une réduction irréversible de la compétence du système et les premières politiques de suppression de cas ne prennent pas ceci en considération.

1. *Politiques de suppression qui préservent la compétence*
   B. Smyth. et M.T. Keane. (Smyth et Keane, 1995b) ont examiné les stratégies de suppression de cas dans le contexte des systèmes de RàPC. Ils ont catégorisé les cas d'un système de RàPC selon leurs compétences. Les concepts clés de cette catégorisation sont le recouvrement (coverage) et l'atteignabilité (reachability).

   - L'ensemble de recouvrement d'un cas représente l'ensemble de cas cibles qu'il peut résoudre.
   - L'ensemble d'atteignabilité d'un cas cible est l'ensemble de cas qui peuvent être utilisés pour le résoudre.

   Quatre classes de cas sont considérées (voir la figure 2.4) :

   (a) *Les cas pivot (pivotal cases)*
   Un cas est dit pivot si son ensemble d'atteignabilité est réduit à un singleton (lui même). Sa suppression réduit directement la compétence du système.

   (b) *Les cas auxiliaires (auxiliary case)*
   Un cas est dit auxiliaire si son recouvrement est subsumé par le recouvrement d'un autre cas. Il n'affecte pas du tout la compétence du système.

   (c) *Les cas de couverture (spanning case)*
   Un cas est dit de couverture si son espace de recouvrement rencontre des régions dans les espaces de recouvrement des cas au sein de son ensemble d'atteignabilité. Ce cas

n'affecte pas directement la compétence du système. Si les cas qui rencontrent des régions de son recouvrement sont supprimés, alors il devient nécessaire pour préserver la compétence.

(d) *Les cas de support (support case)*
Un cas est dit de support s'il existe dans son ensemble d'atteignabilité des cas qui ont le même recouvrement. Ces cas existent en groupe, et la suppression d'un sous-ensemble propre à ce groupe n'affecte pas la compétence. La suppression du groupe tout entier est analogue à celle d'un cas pivot.

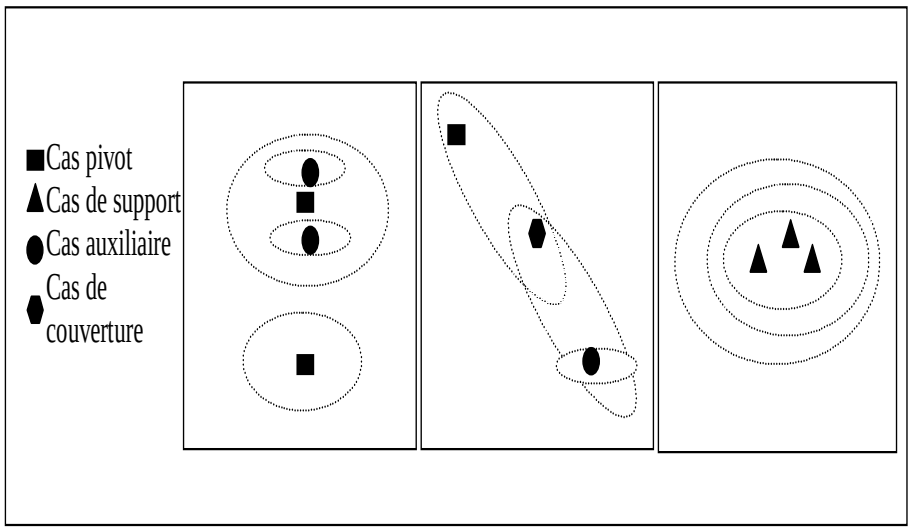


FIG. *2.4: représentation des catégories de cas*

Idéalement, la politique de suppression doit supprimer les cas non pertinents en guidant la base de cas vers une configuration optimale de cas (qui maximise la compétence en minimisant la taille). Cette base de cas optimale est appelée *trace de compétence* (*competence footprint*). Elle procure la même compétence que la base de cas entière avec moins de cas. Les catégories de cas décrites donnent un ordre à respecter dans un ensemble de cas pour pouvoir les supprimer selon leurs contributions à la compétence : les cas auxiliaires sont les moins importants car ils ne contribuent pas directement à la compétence, puis viennent les cas de support. Ensuite, il y a les cas de couverture et enfin les cas pivot qui sont les plus importants. Ceci est à la base d'une nouvelle politique de suppression proposée dans (Smyth et Keane, 1995b) : stratégie de *suppression de traces* (*footprint deletion* : FD).

La politique proposée est destinée à préserver la compétence et ne s'occupe pas du problème

de la performance du système. Pour remédier à ceci, les auteurs proposent une nouvelle stratégie qui combine la politique (FD) avec l'une des politiques qui préserve la performance (citées précédemment). Ceci a donné naissance à la stratégie hybride *suppression trace-utilité* (*footprint-utility deletion* : FUD). Son principe est très simple : d'abord la méthode FD est utilisée pour sélectionner un ensemble de candidats pour la suppression. Ensuite la mesure de performance est utilisée pour choisir le candidat ayant la plus petite performance. Cette nouvelle politique a été proposée dans le but de résoudre le problème de la compétence et celui de la performance.

REMARQUES. Cette approche, bien qu'elle améliore les travaux précédents, possède plusieurs inconvénients, parmi lesquels nous pouvons citer :

- L'hypothèse sur laquelle est fondée l'approche à savoir que « les cas dans la base de cas représentent un échantillon représentatif des problèmes cible » est une hypothèse assez forte et peut ne pas être vérifiée dans plusieurs applications réelles.
- La catégorisation des cas (recouvrement, atteignabilité) est très coûteuse. Yang et Zhu (Yang et Zhu, 2001) ont montré que les algorithmes FD et FUD peuvent perdre presque toute la compétence dans le pire des cas. Donc cette stratégie de suppression ne garantit pas la préservation de la compétence.

Pour améliorer la statégie de suppression de traces, les modèles de compétence définis précédemment (voir l'exemple du paragraphe §2.2.2) ont été utilisés pour effectuer des suppressions de cas à partir de la base de cas (Smyth et McKenna, 1998). Les cas ayant les recouvrements ou les recouvrements relatifs les plus petits sont ceux qui seront supprimés en premiers.

2. *Détection de redondances et d'inconsistances dans une base de cas*
   Le fait que les bases de cas soient souvent construites et mises à jour par différentes personnes à partir de sources de connaissances variées, rend la probabilité qu'elles contiennent des connaissances inconsistantes et redondantes très élevée.
   K.Racine et Q.Yang (Racine et Yang, 1996) et (Racine et Yang, 1997) ont proposé des méthodes et un système pour la maintenance de grandes bases de cas non structurées et semi-structurées. Ils se sont intéressés à deux problèmes dans la maintenance de la base de cas : la suppression de la redondance et de l'inconsistance. Ils ont proposé un système qui interagit avec l'utilisateur en ajoutant deux modules supplémentaires qui sont le module de détection d'inconsistance et celui de détection de redondance.
   Ce qui est intéressant dans leur approche pour la maintenance d'une base de cas de manière générale est leur façon d'effectuer les suppressions pour réduire la base de cas (voir la figure 2.5).

A chaque fois que le module de détection de redondance reçoit un nouveau cas, il détermine s'il est redondant dans la base de cas actuelle. Les cas redondants détectés seront présentés à l'utilisateur. Celui-ci peut alors choisir d'ignorer l'alerte ou de supprimer l'un des cas. Si le nouveau cas n'est pas redondant ou si l'alerte est ignorée par l'utilisateur, il est soumis au module de détection d'inconsistances qui détermine les inconsistances avec d'autres cas de la base de cas. Si une inconsistance est détectée, une alerte est générée pour l'utilisateur pour qu'il fasse une action. Cette alerte identifie le cas concerné et l'endroit où l'inconsistance se produit dans le cas. Si le cas est consistant, il est ajouté à la base de cas, un nouveau test de redondance est effectué par le module de détection de redondance pour tous les cas de la base de cas et une alerte est générée pour l'utilisateur si un cas de redondance est détecté (ce deuxième test de redondance est effectué car des cas de la base de cas peuvent être subsumés par le nouveau cas ajouté).

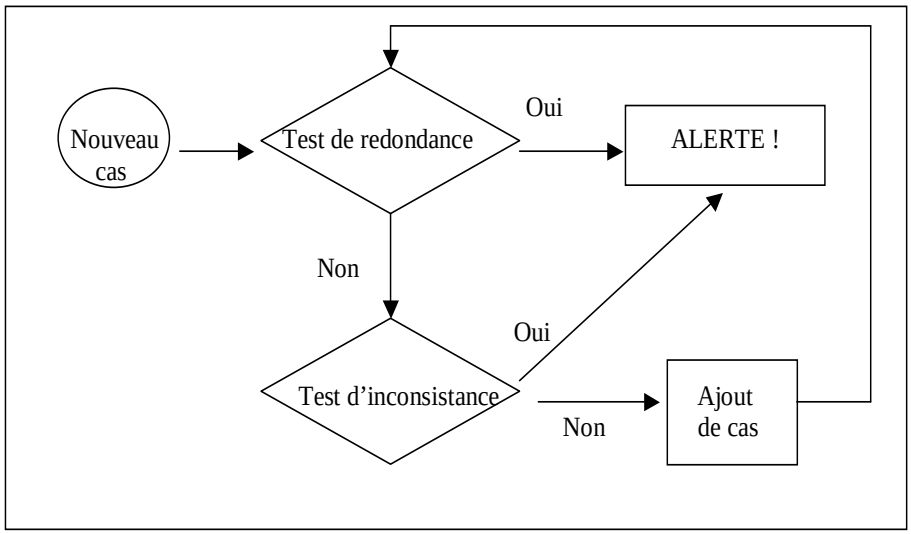


FIG. *2.5: Schéma de construction de la base de cas*

3. *Suppression de cas en utilisant l'induction par arbre de décision floue.*
S. Chi Keung Shiu et al. (Shiu *et al.*, 2001) ont proposé une méthodologie de maintenance des systèmes de RàPC en utilisant l'induction par arbre de décision floue. Cette méthodologie est essentiellement fondée sur l'idée qu'une grande base de cas peut être transformée en une petite base en l'associant à un groupe de règles d'adaptation, qui sont générées par un arbre de décision flou. L'approche proposée intègre l'identification des caractéristiques pertinentes, distingue différents concepts, apprend les connaissances d'adaptation, calcule la compétence de ces cas et sélectionne des représentants de cas regroupés dans une structure de MBC.

Dans la base de cas considérée, toutes les caractéristiques sont à valeurs numériques réelles (ces valeurs peuvent être étendues, sans difficulté à des valeurs dans un espace vectoriel normé). Chaque cas peut être identifié par un index des caractéristiques correspondantes. De plus à chaque cas est associée une action.
Deux nouvelles mesures qui sont *l'ϵ-recouvrement et l'ϵ-atteignabilité* ont été proposées par les auteurs. L'ϵ-recouvrement d'un cas représente la capacité de généralisation de ce cas et l'ϵ-atteignabilité d'un cas représente le degré par lequel ce cas peut être remplacé par un autre cas.
La méthodologie est décrite par les quatre phases suivantes :

- Apprentissage des poids des caractéristiques
- Partitionnement de la base de cas en plusieurs groupes
- Exploitation des règles d'adaptation par un arbre de décision floue
- Suppressions de cas en utilisant l'ϵ-recouvrement et l'ϵ-atteignabilité.

Stratégie d'ajout de cas

Cette stratégie diffère de la première du fait qu'une nouvelle base de cas réduite est construite à partir de la base de cas volumineuse en choisissant les meilleurs cas.

1. *Mesures de la compétence*
   En se basant sur la notion de recouvrement relatif (voir l'exemple du paragraphe §2.2.2), Smyth et McKenna (Smyth et McKenna, 1999) ont défini un nouvel algorithme hybride qui construit une nouvelle base de cas réduite à partir de la base de cas du système. Ils ont proposé de construire cette base de cas en utilisant l'algorithme du plus proche voisin compressé (CNN) sur des cas qui sont initialement ordonnés selon leur valeur de recouvrement relatif. Ceci permet de sélectionner en premier les cas ayant une grande contribution au recouvrement de la base de cas.
   Q.Yang et J. Zhu (Yang et Zhu, 2001) ont proposé une stratégie d'ajout de cas pour construire une base de cas réduite à partir d'une grande base de cas.
   Le recouvrement d'un cas est déterminé par une mesure de similarité et des coûts d'adaptation. Le recouvrement d'un cas est considéré comme le voisinage du cas dans certaines limites d'adaptabilité.

2. *Mesure de la performance*
   Leake et Wilson (Leake et Wilson, 2000) se sont inspirés des travaux de Smyth et McKenna (Smyth et McKenna, 1998) pour définir une mesure de performance relative. La valeur de la performance relative (RP) d'un cas reflète sa contribution à la performance d'adaptation comparée à celle des autres cas. Pour chaque cas candidat à être ajouté à la base de cas, sa

contribution à la performance d'adaptation est estimée. Un ensemble de mesures incluant la mesure du bénéfice de performance produit par l'ajout de chaque cas est estimé. Cependant, les meilleurs résultats sont obtenus en considérant une performance d'adaptation relative. Cette mesure est utilisée pour guider l'ajout de cas en favorisant les cas ayant une grande valeur de RP.

2.2.3.2 Stratégies basées sur le partitionnement de la base de cas

Ces stratégies de maintenance consistent à garder tous les cas de la base de cas et à partitionner celle-ci pour construire plusieurs petites bases de cas. Le but de ces stratégies est de réduire l'espace de recherche de cas dans la base de cas, ce qui réduit aussi le temps de recherche.

Politique de regroupement

Au lieu de réduire la base de cas, Q. Yang et J. Wu (Yang et Wu, 2000) ont proposé une autre stratégie qui consiste à garder tous les cas (il n'y a pas de perte d'information) et à diviser la grande base de cas en groupes formant de petites bases de cas où la recherche de cas n'est pas coûteuse.

La stratégie de maintenance proposée par Q. Yang et J. Wu (Yang et Wu, 2000) est effectuée pour assurer l'efficacité de la maintenance de la base de cas. Elle est basée sur deux idées principales :

- La première est la partition : une grande base de cas est décomposée en groupes constitués de cas proches. Une collection de bases de cas distribuées est ainsi créée.
- La deuxième consiste à permettre à un utilisateur de rechercher les bases de cas distribuées en sélectionnant de manière incrémentale les attributs qui sont riches en informations et qui peuvent couvrir la structure de la base de cas entière. Ces attributs sont présentés à l'utilisateur de manière interactive.

Des petites bases de cas sont construites à partir du résultat du regroupement. Chaque petite base possède un nom qui représente la description de cette base et un ensemble d'attributs associés aux cas. A chaque attribut est associé un poids qui représente la moyenne des poids des cas du groupe.

Stratégies de partitionnement utilisant les réseaux de neurones

Plusieurs approches de RàPC qui utilisent les réseaux de neurones pour indexer la base de cas ont été proposées dans la littérature. La base de cas est alors divisée en plusieurs parties. Chaque partie est obtenue par regroupement de cas similaires effectué par le réseau de neurones. Nous pouvons distinguer les systèmes qui utilisent des réseaux de neurones évolutifs qui prennent en compte l'arrivée dynamique des données (Malek, 2000) et (Fdez-Riverola et Corchado, 2003) de ceux qui utilisent des réseaux de neurones statiques (Mujica et Vehi, 2003) et (Roh *et al.*, 2003).

Ces systèmes sont décrits de manière détaillée dans le chapitre 6.

2.3 Discussion

L'approche proposée par Leake et Wilson permet de catégoriser les approches de maintenance de la base de cas en déterminant quand et comment un système de RàPC exécute la MBC. Par exemple, l'approche de suppression proposée par Smyth et Keane (Smyth et Keane, 1995b) consiste à classer les cas en termes d'atteignabilité et de recouvrement, puis à sélectionner un candidat pour la suppression une fois que la taille de la base de cas dépasse une certaine limite prédéfinie. Le classement est défini en respectant l'état courant de la base de cas. Comme elle est déclenchée comme réponse à la taille courante de la base de cas, le moment de déclenchement est conditionnel. Ce mécanisme est appliqué soit à un petit nombre de cas durant le processus, utilisant une méthode heuristique d'évaluation d'utilité (FUD) (portée restreinte) ou à un grand nombre de cas en dehors du cycle (hors ligne et de portée générale).
Les travaux cités précédemment ont abordé des notions importantes pour la maintenance de la base de cas qui sont :

- *La performance :* temps mis par le système pour trouver une solution au cas cible.
- *La compétence :* recouvrement de la base de cas.
- *La mise à jour des cas :* effectuée quand l'environnement est dynamique.

La mise à jour des cas de la base de cas comporte deux aspects :

- Le parcours de la base de cas pour détecter des anomalies (par exemple : l'inconsistance et la redondance (Racine et Yang, 1996) et (Racine et Yang, 1997))
- Le changement de la structure du cas cible avec le changement du domaine (par exemple : le changement des poids des caractéristiques pertinentes des cas (Shiu *et al.*, 2000) et (Shiu *et al.*, 2001))

Les notions de performance et de compétence ont été étudiées dans la plupart des travaux que nous avons cités. Les auteurs de ces travaux ont proposé des approches qui améliorent la performance du système tout en gardant une compétence raisonnable.
Certaines approches se sont intéressées à la réduction de la base de cas quand sa taille ne peut dépasser une certaine limite. La notion de recouvrement a été beaucoup étudiée et définie de différentes manières par les auteurs. Les cas supprimés de la base de cas (ou non ajoutés à la base de cas) sont ceux qui contribuent le moins à la compétence de la base de cas.
D'autres approches comme celle proposée dans (Yang et Wu, 2000) se sont intéressées à la réduction de l'espace de recherche pour améliorer la performance tout en gardant tous les cas de la base de cas.
Nous nous intéressons à la maintenance d'une base de cas constituée d'un grand nombre de sé-

quences d'un système de RàPC dans un domaine qui contient du bruit. Les trois notions citées ci-dessus sont toutes importantes pour notre problème.
Les travaux cités visent principalement à améliorer la performance des systèmes de RàPC et n'abordent l'amélioration de la compétence qui peut être détériorée par la présence du bruit dans les cas comme dans le cas de notre application. De plus, ces travaux ne traitent pas des séquences, donc n'ont pas abordé les problèmes liés au fait que les cas sont des successions d'événements et non des attributs-valeurs. Ces problèmes concernent le contenu de la mémoire : base de séquences et/ou base de cas, en fonction de la représentation d'un cas dans le système de RàPC. En plus du contrôle du contenu de la base de cas, la gestion du fonctionnement du système de RàPC s'impose. Quand un cas, par exemple, est une expérience précise dans une séquence qui représente une succession d'instants comme dans notre système, le choix de construire une base de cas et une base de séquences nous est semblé le meilleur pour améliorer la performance de notre système en favorisant l'utilisation de la base de cas le plus souvent possible pour éviter l'extraction de cas à partir de séquence. De plus, la présence de la base de séquences dans la mémoire permet de préserver la compétence du système et d'enrichir à chaque fois la base de cas.

Nous allons d'abord décrire notre système de RàPC ainsi que notre approche pour la maintenance de la base de cas, puis nous allons les comparer aux travaux que nous avons cités précédemment.

Chapitre 3

Système de RàPC pour la prédiction de l'évolution d'une séquence

Dans ce chapitre, nous proposons un nouveau système de RàPC baptisé « CASEP : CAse-based reasoning system for SEquence Prediction » qui effectue le classement (ou la prédiction) de séquences (Zehraoui *et al.*, 2003) et (Zehraoui, 2003). Une description détaillée de CASEP ainsi que les expérimentations réalisées pour sa validation sont présentées.

3.1 Description Générale

Nous commençons d'abord par donner quelques définitions des termes qui seront utilisés par la suite.
Une séquence $q = (E_j^q)$, $1 \leq j \leq m(q)$ est une succession ordonnée de $m(q)$ états E_j^q. Par exemple, dans notre application, une séquence représente une navigation complète d'un utilisateur d'un site Web.
Un état $E_j^q = (v_i)$, $1 \leq i \leq n$ de la séquence q est caractérisé par un ensemble de valeurs $v_i (1 \leq i \leq n)$ prises par n variables c_i $(1 \leq i \leq n)$ appelées caractéristiques et par la position j de E dans la séquence q.
Le nombre $m(q)$ représente la longueur de la séquence q.
Dans notre application, un état peut représenter une ou plusieurs pages ayant des caractéristiques communes, des exemples de ces caractéristiques sont : l'URL de la page, sa taille, etc.
Notre problème consiste à prédire une valeur $s_j (1 \leq j \leq l)$ d'une propriété S d'un ensemble d'états succédant l'état courant d'une séquence q (cette propriété peut représenter, par exemple, une caractéristique de l'état suivant ou la classe de la séquence courante et l représente le nombre de valeurs possibles de S).
Pour ceci, nous disposons d'un grand nombre de séquences de longueurs variables qui représentent des comportements enregistrés dans le passé.

Description de la structure d'un cas

Un cas cible représente les p derniers états de la séquence courante (figure 3.1) où p est un paramètre du système.
Nous cherchons à prédire la valeur d'une propriété $s_j \in S$ $(1 \leq j \leq l)$ d'un ensemble d'états qui succèdent à l'état courant (cette propriété peut être la classe de la séquence courante : dans notre application cette classe représente le profil de l'utilisateur { acheteur, non acheteur }). Pour ceci, nous effectuons une recherche dans la mémoire de notre système qui est constituée d'une base de cas et d'une base de séquences. Des cas sources sont remémorés à partir de la base de cas ou extraits à partir de la base de séquences.
Un cas source représente une expérience précise dans une séquence. Il est constitué de deux parties :

- *La partie problème :* représente un morceau d'une séquence de longueur fixée p : c'est un vecteur de p états.
- *La partie solution :* représente une valeur s_j de la propriété S.

Lors du calcul des similarités entre le cas cible et le cas source, les deux cas sont comparés état par état. Dans cette comparaison, l'ordre des états et la présence d'écarts dans la séquence entre deux états successifs du cas sont pris en compte (voir figure 3.1). La longueur de l'écart $e \in N$ permis est un paramètre de notre système.

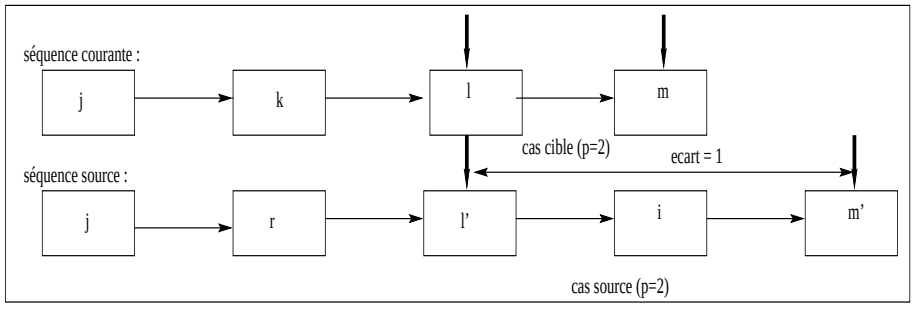


FIG. *3.1: Représentation d'un cas cible et d'un cas source.*

3.2 Approche de la maintenance de la base de cas

Comme dans (Jaczynski, 1998), deux types de cas sont définis :

- Les cas sources potentiels qui sont extraits à partir de la base de séquences.
- Les cas sources concrets qui sont remémorés à partir de la base de cas.

De nouvelles mesures sont associées aux cas concrets. Ces mesures servent à améliorer les résultats du système et à effectuer la maintenance de la base de cas. Elles sont décrites comme suit :

- *La contribution positive CP :* représente la contribution du cas source à effectuer une bonne résolution des nouveaux problèmes (cas cibles).
- *La contribution négative CN :* représente la contribution du cas source à effectuer une mauvaise résolution des nouveaux problèmes (cas cibles).
- *La qualité d'un cas QC :* $QC = \frac{CP}{CN+CP}$ représente le taux d'utilisation du cas avec succès.
- *L'étendue ET :* détermine le voisinage d'un cas source qui contient les cas recouverts par ce cas et varie quand la QC de ce cas diminue. Pour tout cas source cas_i, $ET(cas_i)$ définit un seuil de similarité variable $\alpha(cas_i)$ propre à cas_i : $\alpha(cas_i) = 1 - ET(cas_i)$.

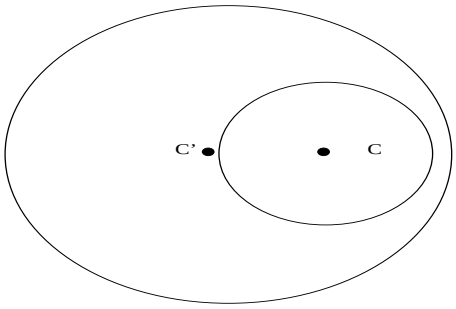


FIG. *3.2: Le recouvrement du cas c est inclus dans celui du cas c'. Si les deux cas ont la même solution, c n'est pas ajouté à la base de cas.*

L'initialisation et la mise à jour de ces mesures sont effectuées dans le cycle du RàPC décrit dans le paragraphe §3.3.
La stratégie de maintenance proposée consiste à :

- Apprendre de nouveaux cas sélectionnés à partir des cas extraits de la base de séquences, ceci permet de rendre les cas sources potentiels concrets (voir la phase d'apprentissage dans le paragraphe §3.3) ;
- Détecter et réduire l'utilisation de « cas-bruit » qui peuvent détériorer les résultats du système en diminuant leurs étendues (ET). Ceci revient à augmenter leurs seuils de similarité. Un cas concret cas_i est dit « cas-bruit » si sa qualité est inférieure à un seuil d' ($QC(cas_i) < d' : 0 < d' < 1$ est un seuil défini dans le système) ;
- Réduire la taille de la base de cas concrets en dehors du cycle de RàPC (hors ligne) (Leake et Wilson, 1998). La notion de recouvrement est utilisée pour construire une base de cas réduite. L'ensemble de recouvrement d'un cas c représente l'ensemble de cas qui sont si-

milaires à c et qui ont la même solution que c (ou qui satisfont certaines conditions. Ces conditions peuvent représenter l'appartenance à un intervalle autour de la solution de c pour la tâche de prédiction.).

La réduction de la base de cas consiste à construire, à partir de la base de cas courante, une base de cas réduite (initialement vide) en lui ajoutant progressivement des cas (stratégie d'ajout de cas). Les cas sont d'abord ordonnés selon leur étendue décroissante. Si deux cas ont la même étendue, ils sont ordonnés par leur qualité décroissante. Un cas c n'est pas ajouté à la base de cas réduite (cas non utile) si c est recouvert par un autre cas c' dans cette base et si le voisinage de c est inclus dans celui de c' (voir la figure 3.5). Ce voisinage représente l'ensemble des cas de la base de cas dont la similarité avec c est supérieure ou égale à $\alpha(c)$. La réduction de la base de cas est effectuée suivant l'algorithme 1.

Algorithme 1 Réduction de la base de cas.

$liste \leftarrow ()$
//liste est un ensemble ordonné de cas
Ordonner les cas de BC dans liste selon leurs ET, puis QC décroissants et les mettre dans $liste$
// construire la base de cas réduite
$NBC \leftarrow \{premier\_element\_de\_liste\}$
//NBC est la nouvelle base de cas réduite
Pour tout cas $cas_i \in liste$
  Pour tout cas $cas_j \in NBC$
  //NBC est la nouvelle base de cas
    Si $[(1 - Similarite(cas_i, cas_j) + ET(cas_i) < ET(cas_j)) \wedge (sol(cas_i) = sol(cas_j))]$
    alors mettre à jour les mesures associées à cas_j
    Sinon $NBC \leftarrow NBC \cup \{cas_i\}$
    finSi
  finPour
finPour

3.3 Architecture du système CASEP

Le système « CASEP : CAse based reasoning for SEquences Prediction » est composé principalement de deux modules (voir la figure 3.3) : Le *moteur naïf* et le *moteur intelligent*. Ces deux modules sont contrôlés par un module appelé *contrôleur*. Lorsqu'un nouveau problème se présente au système, le moteur intelligent déclenche le cycle du raisonnement à partir de cas. Dans ce cycle, les cas sources sont remémorés à partir de la base de cas. Si ce module ne peut pas fournir la solution, alors le moteur naïf est activé. Celui-ci déclenche le cycle du RàPC, mais cette fois, les cas sources sont extraits à partir de la base de séquences. Certains cas extraits

sont ajoutés à la base de cas dans la phase d'apprentissage. La réduction de la base de cas est effectuée régulièrement par le système, en dehors du cycle de RàPC. Le cycle du raisonnement à partir de cas est décrit ci-dessous :

3.3.1 Phase de recherche

La mémoire du système contient deux parties principales :

- La base de séquences qui contient les séquences entières ;
- La base de cas qui contient les cas concrets.

La mesure de similarité utilisée compare les états du cas source à ceux du cas cible tout en tenant compte de l'ordre de ces états ainsi que des écarts pouvant exister entre les états (voir figure 3.1). Les états du cas cible doivent être similaires à ceux du cas source et présentés dans le même ordre.
La recherche de la solution du cas cible est effectuée en remémorant les cas sources dont la similarité au cas cible dépasse un seuil $0 < \alpha < 1$. Ces cas sont remémorés à partir de la base de cas ou extraits à partir de la base de séquences.
La recherche est d'abord effectuée dans la base de cas. Le seuil de similarité $\alpha(ET)$ est une fonction de l'étendue (ET) de chaque cas source et permet de filtrer les cas remémorés. Si aucune solution n'est fournie en utilisant cette base, alors la recherche est effectuée dans la base de séquences. Un ensemble de cas est alors extrait à partir des séquences. Le seuil de similarité, dans ce cas de figure, est constant (α_0) (voir l'algorithme 2).

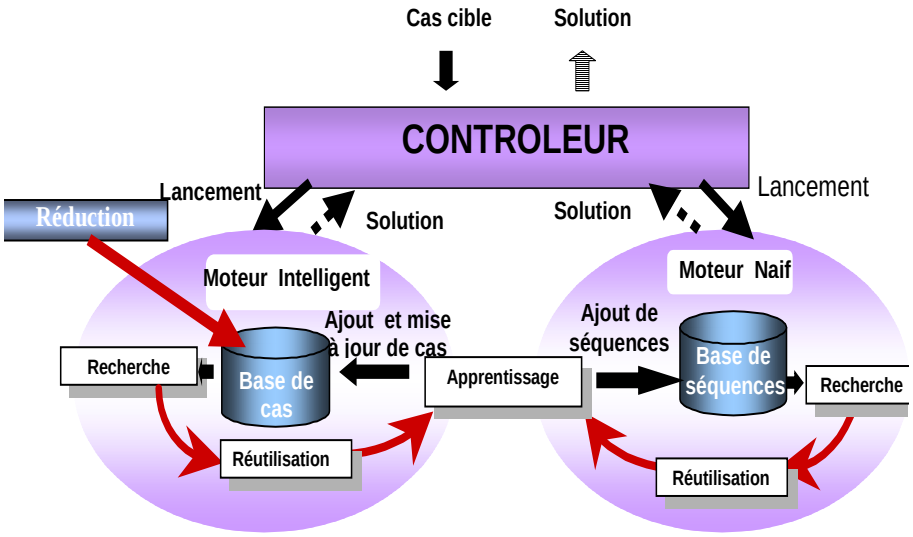


FIG. 3.3: *Architecture du système « CASEP ».*

Algorithme 2 Remémoration de cas.

Pour tout $cas_i \in BC$ //BC est la base de cas
//Recherche dans la base de cas
Si $Similarite(cas_i, cas_{cible}) > 1 - ET(cas_i)$ **alors** cas_i est remémoré
finSi
finPour
Si(aucun cas n'est remémoré) $\vee$ (pas de solution) **alors**
//recherche dans la base de séquences BS
Pour toute séquence q dans BS
Pour tout cas cas_i extrait à partir de q
Si $Similarite(cas_i, cas_{cible}) > \alpha_0$ **alors** cas_j est remémoré
finSi
finPour
finPour
finSi

3.3.2 Phase de réutilisation

Après la phase de recherche, un ensemble de cas sources est remémoré. Pour chaque valeur de la solution s_j de S proposée par ces cas, nous associons une confiance. La confiance associée à s_j est une fonction d'agrégation (algorithme 3) de :

- La moyenne des similarités au cas cible ($Similarite\_moy(s_j)$) des cas remémorés dont la solution est s_j.
- Le pourcentage ($pourcentage(s_j)$) des cas qui proposent s_j parmi l'ensemble des cas remémorés.

La moyenne des similarité associée à s_j ($Similarite\_moy(s_j)$) est calculée de manières différentes suivant le type des cas sources remémorés (cas potentiels extraits à partir de la base de séquences ou cas concrets remémorés à partir de la base de cas). Ceci est dû au fait que les mesures de maintenance ne sont associées qu'aux cas concrets. La qualité (QC) des cas concrets remémorés est utilisée pour calculer une moyenne pondérée des similarités (voir l'algorithme 3). La valeur s_j ayant la plus grande confiance est fournie par le système. Si plusieurs valeurs de S ont des confiances proches (un seuil δ, $0 < \delta < 1$ est défini dans le système) de la plus grande confiance, alors le système ne peut pas résoudre le cas cible.

3.3.3 Phase d'apprentissage

Dans cette phase, deux opérations sont effectuées :

Algorithme 3 Calcul des confiances.

Si les cas remémorés $\in$ BC **alors** //BC est la base de cas
  Pour toute valeur proposée $s_j \in S$
  $Similarite\_moy(s_j) = \frac{1}{\sum_i QC(cas_i)} * \sum_i [QC(cas_i) * Similarite(cas_i, cas_{cible})]$
  $//QC(cas_i)$: qualité de cas_i
    $Conf(s_j) = A * pourcentage(s_j) + B * Similarite\_moy(s_j)$
    //A et B sont des paramètres du système.
  finPour
finSi
Si les cas remémorés $\in$ BS **alors** //BS est la base de séquences
  Pour toute valeur proposée $s_j \in S$
  $Similarite\_moy(s_j) = \frac{1}{|E_j|} * \sum_{cas_i \in E} Similarite(cas_i, cas_{cible})$
E_j est l'ensemble des cas remémorés de solution s_j
    $Conf(s_j) = A * pourcentage(s_j) + B * Similarite\_moy(s_j)$
  finPour
finSi

- L'ajout de séquences à la base de séquences ;
- L'ajout de cas à la base de cas.

L'ajout de séquences à la base de séquences est effectué en utilisant une heuristique simple. Cette heuristique consiste à ajouter toute séquence de longueur supérieure ou égale à p. Sachant que la séquence courante évolue dans le temps (à chaque instant, un nouvel état est formé), seules les séquences entières qui contiennent tous les états sont ajoutées à la base de séquences. Donc l'ajout de séquences ne peut être effectué qu'à la fin de chaque séquence courante.
Avec le temps, le nombre de séquences augmente. Comme la mémoire du système est limitée, les séquences ne peuvent être toutes retenues. Une autre heuristique est utilisée pour supprimer des séquences de la base de séquences (dans notre application, nous avons choisi de supprimer les navigations les plus anciennes).
Les cas ajoutés à la base de cas sont des cas potentiels extraits à partir de la base de séquences. Pour éviter de garder tous les cas extraits à chaque cycle de RàPC (ceci produit une base de cas volumineuse), nous avons utilisé une heuristique pour sélectionner les cas représentatifs à partir de ces cas. Cette heuristique consiste à :

- Effectuer un regroupement dans l'ensemble des cas extraits qui possèdent la même solution que celle du cas cible. Nous avons utilisé l'algorithme des matrices de similarité (Fu, 1992) qui utilise seulement les similarités entre les cas.
  Cet algorithme transforme, d'abord, la matrice de similarité en une matrice transitive équivalente, ensuite, détermine des groupes en se basant sur la règle : « *deux cas c et c' appartiennent au même groupe si et seulement si similarite$(c, c') \geq \alpha_0$, où α_0 est le seuil*

de similarité[5] *défini précédemment* ».
- Sélectionner un représentant de chaque groupe (ce représentant est le cas qui maximise la somme des similarités aux autres cas du même groupe, c'est le plus proche du barycentre).

Les représentants sélectionnés sont ajoutés à la base de cas.

Initialisation des mesures de qualité :
A chaque fois qu'un cas est ajouté à la base de cas, des mesures de qualité lui sont associées et sont initialisées comme suit :
$CN = 0$, $CP = \frac{1}{nb'}$ et $ET = 1 - \alpha_0$
où nb' est le nombre des représentants ajoutés et α_0 est le seuil de similarité initial.

Mise à jour des mesures de qualité
Quand la solution fournie par le système est obtenue à partir des cas concrets, des mises à jour sont effectuées pour les mesures de qualité associées à ces cas concrets (algorithme 4).

- Si la solution fournie par le système est correcte, alors les contributions positives CP des cas remémorés dont la partie solution est celle fournie par le système sont augmentées.
- Si la solution fournie par le système n'est pas correcte, alors les contributions négatives CN des cas dont la partie solution est celle fournie par le système sont augmentées. Si la qualité QC de l'un de ces cas est inférieure à un certain seuil d', alors son étendue ET est diminuée. Ceci permet de diminuer l'utilisation des cas bruits en augmentant leur seuil de similarité.

Ces mises à jour ne peuvent être effectuées qu'après la connaissance de la solution du cas cible. Par exemple, dans notre application, la classe de la navigation n'est connue qu'à la fin de la navigation, et ce n'est qu'à ce moment là que les mises à jour et les ajouts de cas à la base de cas peuvent être effectués. Donc tous les traitements effectués sur la navigation courante sont mémorisés jusqu'à la fin de la navigation.

REMARQUE. Les quantités par lesquelles les mesures CP et CN sont augmentées peuvent dépendre des qualités et des similarités des cas sources remémorés qui ont contribué à la résolution du cas cible.

3.3.4 Phase de réduction

Cette phase est effectuée en dehors du cycle du raisonnement à partir de cas. Elle est déclenchée périodiquement après un certain nombre de séquences. La réduction de la base de cas

[5] Nous avons choisi ce seuil puisque c'est celui que nous avons défini pour dire que deux cas sont similaires ou pas.

Algorithme 4 Mise à jour des mesures de qualité d'un cas.

Pour tout cas cible cas_i
Si $ValSol(cas_i)$ est fournie par BC **alors** //BC est la base de cas
Pour tout cas cas_j remémoré de BC
Si $valSolution(cas_i) = sol(cas_j)$ **alors**
// cas_j a contribué à la résolution de cas_i // $valSolution(cas_i)$ est la solution du cas cible cas_i fournie par le système
Si $valSol(cas_i) = sol(cas_i)$ **alors** // Résolution correcte du cas cible
$CP(cas_j) = CP(cas_j) + \frac{1}{nb}$
// nb est le nombre de cas remémorés ayant contribué à la résolution du cas cible
Sinon
$CN(cas_j) = CN(cas_j) + \frac{1}{nb}$
Si $QC(cas_i) < d'$ **alors** $ET(cas_i) \leftarrow pr.ET(cas_i)$
// d' est le seuil défini précédemment et $0 < pr < 1$ est le pourcentage par lequel l'étendue est diminuée.
finSi
finSi
finSi
finPour
finSi
finPour

décrite précédemment (voir le paragraphe §3.2) est appliquée au système.

3.4 Étude Comparative

Nous avons proposé une stratégie de maintenance d'une base de cas d'un système de RàPC dont la mémoire est constituée de deux bases :

- La base de séquences qui contient des séquences entières.
- La base de cas qui comporte des expériences précises extraites de ces séquences comme dans (Jaczynski, 1998).

Contrairement aux cycles de RàPC classiques, les cas ajoutés à la base de cas ne sont pas des cas cibles : ce sont des cas extraits de la base de séquences. De plus, plusieurs cas peuvent être ajoutés à la base de cas durant un cycle de RàPC.
Dans un premier temps, nous avons sélectionné les cas ajoutés à la base de cas en effectuant des regroupements sur l'ensemble des cas extraits de la base de séquences et en choisissant un représentant de chaque groupe.
Nous avons associé aux cas des mesures pour effectuer la maintenance. Des mises à jour de ces

mesures de maintenance sont effectuées à chaque cycle du RàPC[6]. Ces mesures permettent de déterminer la pertinence d'un cas suivant son utilisation dans la résolution des cas cibles :

- L'étendue d'un cas détermine un voisinage du cas qui contient les cas qui peuvent être recouverts par ce cas. Mais ce voisinage peut contenir des cas de solutions différentes. L'ensemble de recouvrement d'un cas est l'ensemble des cas qui sont dans son voisinage et qui ont la même solution que ce cas (ou ont des solutions proches dans le cas de la prédiction : un intervalle autour de la solution du cas cible peut être défini) ;
- La qualité d'un cas détermine le taux de réussite du cas à résoudre les cas cibles qui sont présentés au système ;
- Les contributions positive et négative mesurent la fréquence d'utilisation des cas avec succès et échec pour résoudre les cas cibles tout en tenant compte du nombre de cas qui participent à la résolution de chaque cas.

La notion de recouvrement définie par Smyth et McKenna (Smyth et McKenna, 1998) et (Smyth et McKenna, 1999) consiste à considérer le recouvrement d'un cas cas_i comme étant l'ensemble des voisins de cas_i (selon la mesure de similarité définie) tel que la solution de cas_i puisse être adaptée pour résoudre les cas de cet ensemble. Dans (Yang et Zhu, 2001) l'ensemble de recouvrement d'un cas cas_i représente le voisinage d'un cas suivant le coût d'adaptation (ensemble de cas dont la solution est proche de celle de cas_i).
Le recouvrement d'un cas tel qu'il est défini dans notre approche est défini de la même manière que dans (Smyth et McKenna, 1998) et (Smyth et McKenna, 1999) sauf que la notion d'adaptation est différente. Les cas qui recouvrent réellement le cas cible sont ceux qui ont la même solution parmi ces cas (on s'intéresse à un problème de classement).
Généralement, la régularité problème-solution est posée comme hypothèse dans les approches de RàPC. Cette régularité représente la relation entre les mesures de similarité des problèmes et celles des solutions qui garantit que des problème similaires ont des solutions similaires (Leake et Wilson, 1999). Dans notre approche, nous supposons que l'hypothèse de régularité problème-solution est vérifiée en moyenne : c'est à dire qu'elle est vérifiée par la majorité des cas, ce qui justifie l'utilisation du RàPC, mais ceci n'exclut pas la présence de « cas-bruit ». Par conséquent, tout cas similaire au cas cible est un candidat potentiel à sa résolution.
Pour ceci, la phase d'adaptation consiste à donner une solution au nouveau cas cible à partir d'un ensemble de cas en tenant compte des similarités de ces cas au cas cible et de leurs représentativités. Dans les systèmes de RàPC, généralement le cas le plus similaire au cas cible est remémoré. Mais dans notre étude, comme nous l'avons déjà mentionné, l'environnement est complexe et dynamique et le choix du cas le plus similaire au cas cible est difficile. Pour ceci, nous avons choisi de remémorer un ensemble de cas cibles au lieu d'un seul. De plus les notions d'étendue et de qualité de cas permettent de faire face au problème des cas-bruit.
Les notions de contributions positives et négatives pourront aussi nous permettre de supprimer

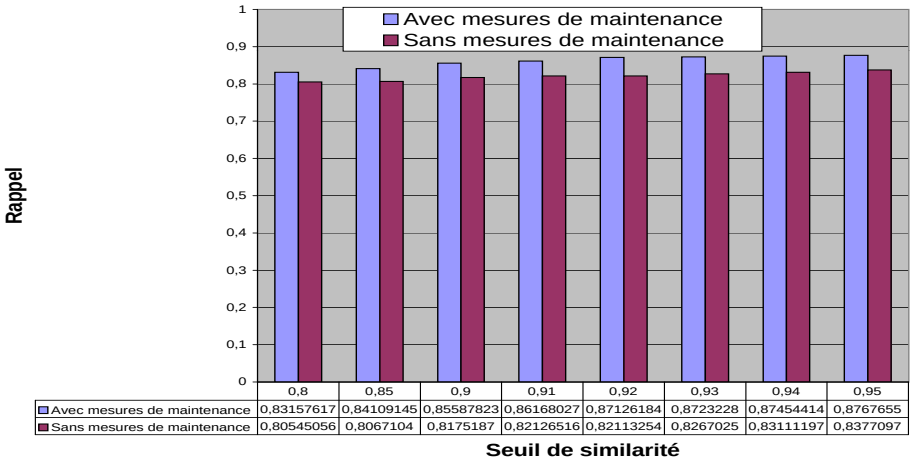
[6] Ces mises à jour sont retardées jusqu'à ceque la solution du cas cible soit connue.

les cas obsolètes (ce qui n'a pas été pris en compte pour l'instant dans notre approche). Ces cas sont ceux qui ne sont pas utilisés par le système pour fournir des solutions aux cas cible.
Les notions liées au coût d'adaptation ne peuvent être utilisées dans notre approche puisque nous ne construisons pas la solution.
L'idée d'utiliser les notions d'échec et de succès pour déterminer les cas à retenir dans la base de cas a été proposée dans (Portinale *et al.*, 1999). Mais dans cette approche, les cas potentiellement dangereux (qui échouent dans la phase d'adaptation) ou les cas non utilisés (qui ne sont pas remémorés pendant une longue période de temps) sont directement supprimés, tandis que dans notre approche, l'utilisation des cas-bruit est progressivement réduite et les cas non utiles sont supprimés. Les notions de succès et d'échec définies dans notre approche sont différentes de celles définies dans (Portinale *et al.*, 1999), où le succès et l'échec concernent l'adaptation de cas (dans une situation d'échec, le système de RàPC ne peut pas fournir de solution), tandis que dans notre approche, le succès et l'échec sont liés à la qualité de la solution fournie par le système. Dans une situation d'échec, le système fournit une mauvaise solution (ceci est dû à la présence des cas-bruit).
L'approche Broadway (Jaczynski, 1998) utilise la même représentation de cas que dans notre système (un cas est une partie d'une séquence). Dans CASEP, l'étendue ET est utilisée pour réduire l'utilisation des cas-bruit en augmentant leur seuil de similarité. Dans le système Broadway-V1, le seuil de similarité est constant pour tous les cas, tandis que dans notre approche, il est variable et propre à chaque cas. Dans Broadway-V1 (Jaczynski et Trousse, 1998), une stratégie de maintenance a été proposée. Les mesures ci-dessous sont associées aux cas concrets :

- Le nombre de remémoration de cas avec succès
- La qualité du cas = $\frac{nombre\_rememorations\_succes}{nombre\_rememorations}$

Ces mesures permettent d'améliorer la prédiction (utilisation de la qualité de cas dans la phase de réutilisation) et de réduire la base de cas.
Le nombre de remémorations avec succès est proche de la contribution positive (CP) du cas, mais dans notre mesure, nous prenons en compte le nombre de cas qui permettent de fournir la solution du cas cible (quand plusieurs cas sont utilisés avec succès pour fournir la solution du cas cible, les CP de ces cas sont augmentés de manière différente que quand un seul cas fournit la solution). La définition de la qualité d'un cas est similaire à la notre, mais dans notre mesure, nous tenons toujours compte de la contribution de cas à résoudre le cas cible. Nous avons aussi proposé d'autres mesures qui sont la contribution négative CN et l'étendue ET.
La réduction de la base de cas dans Broadway-V1 (Jaczynski et Trousse, 1998) est limitée à la suppression des cas qui ne sont pas utilisés pendant une longue période (cas obsolètes). Dans notre approche, nous étudions l'état de cette base et les similarités qui existent entre les cas de la base de cas (recouvrement) en utilisant les qualités des cas QC et leurs étendues ET.

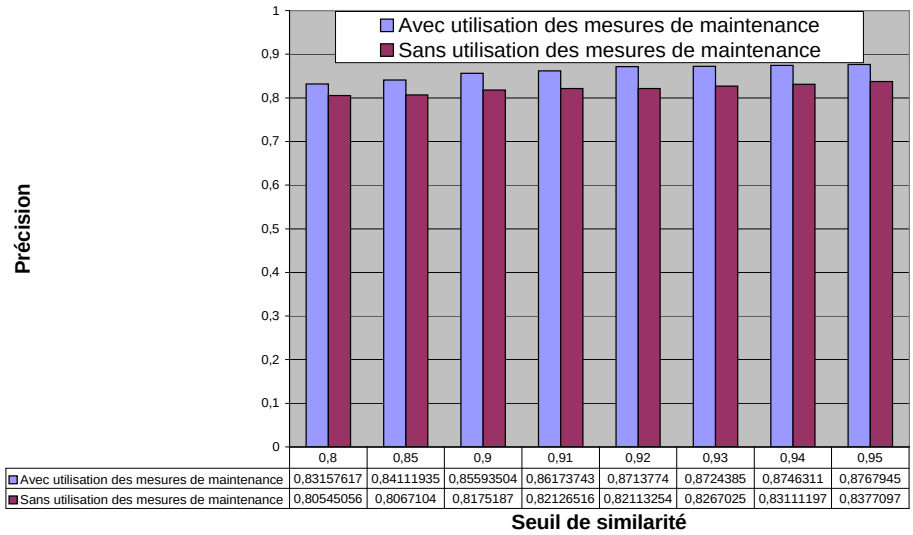
3.5 Validation de l'approche proposée



	0,8	0,85	0,9	0,91	0,92	0,93	0,94	0,95
Avec mesures de maintenance	0,83157617	0,84109145	0,85587823	0,86168027	0,87126184	0,8723228	0,87454414	0,8767655
Sans mesures de maintenance	0,80545056	0,8067104	0,8175187	0,82126516	0,82113254	0,8267025	0,83111197	0,8377097

FIG. *3.4: Comparaison du rappel en utilisant la base d'apprentissage.*

Dans ce paragraphe, nous allons d'abord décrire les expérimentations que nous avons réalisées sur une grande base de données pour valider les mesures de maintenance que nous avons proposées. Ensuite, nous allons décrire les expérimentations effectuées sur notre système de prédiction dans le cadre de l'application que nous avons citée.

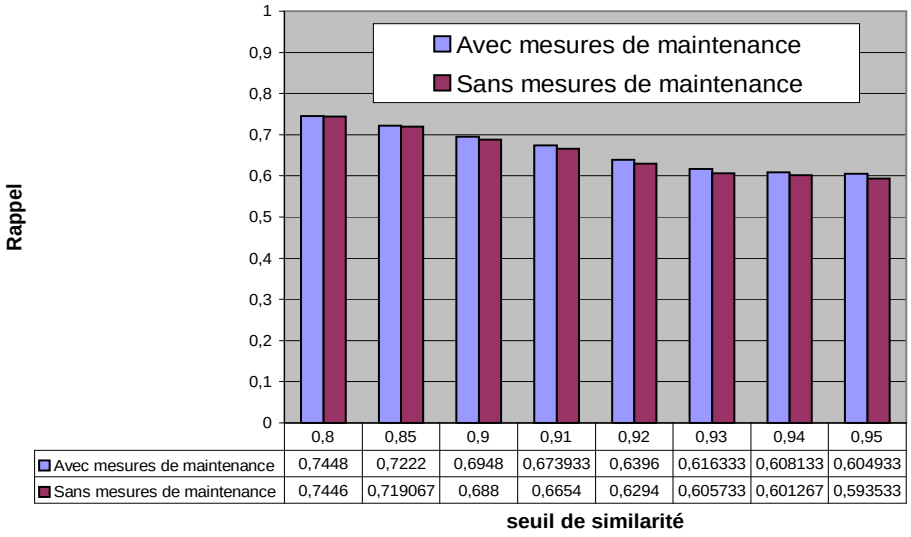


	0,8	0,85	0,9	0,91	0,92	0,93	0,94	0,95
Avec utilisation des mesures de maintenance	0,83157617	0,84111935	0,85593504	0,86173743	0,8713774	0,8724385	0,8746311	0,8767945
Sans utilisation des mesures de maintenance	0,80545056	0,8067104	0,8175187	0,82126516	0,82113254	0,8267025	0,83111197	0,8377097

FIG. *3.5: Comparaison de la précision en utilisant la base d'apprentissage.*

3.5.1 Tests effectués sur une grande base de données pour valider les mesures de maintenance

Nous avons effectué des expérimentations sur une base de données pour le classement des salaires de personnes adultes « base de données adult » : [*http ://ftp.ics.uci.edu/pub/machine-learning-databases/adult/*]
Cette base contient 45222 instances complètes (30162 instances d'apprentissage et 15060 instances de test). Chaque instance contient 14 attributs (6 continus et 8 symboliques). Chaque cas est constitué d'une partie problème qui est décrite par les 14 caractéristiques, une partie solution qui représente la classe de la personne selon son salaire ($\leq 50K$ *et* $> 50K$) et les mesures de qualité décrites précédemment.



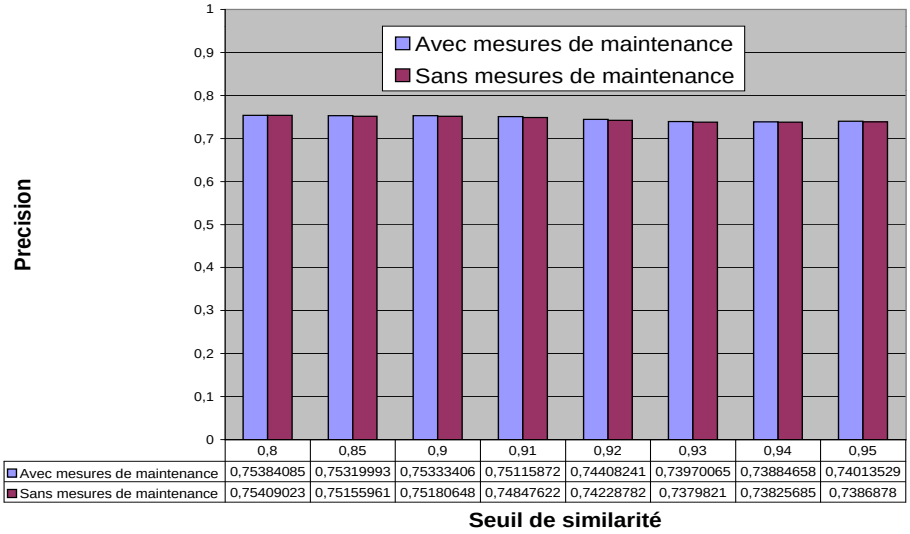
	0,8	0,85	0,9	0,91	0,92	0,93	0,94	0,95
Avec mesures de maintenance	0,7448	0,7222	0,6948	0,673933	0,6396	0,616333	0,608133	0,604933
Sans mesures de maintenance	0,7446	0,719067	0,688	0,6654	0,6294	0,605733	0,601267	0,593533

FIG. *3.6: Comparaison du rappel en utilisant la base de test.*

Ces expérimentations sont effectuées pour étudier l'effet de l'utilisation des mesures associées au cas de la base de cas (qualité de cas et diminution de l'étendue) sur les résultats fournis par le système de RàPC. Pour ceci, nous comparons les résultats du système en utilisant deux approches. Dans la première, les mesures associées au cas sont utilisées. Dans la deuxième, ces mesures ne sont pas utilisées. Ces approches diffèrent principalement dans les phases de réutilisation et d'apprentissage.
Les mesures utilisées pour comparer les deux approches sont :

- *La précision :* représente le taux de bons classements parmi les classements proposées par le système ($\frac{nombre\_bon\_classements}{nombre\_classements}$).
- *Le rappel :* représente le taux de bons classements parmi tout les classements que le système

devrait effectuer ($\frac{nombre\_bon\_classements}{nombre\_cas\_cibles}$).



	0,8	0,85	0,9	0,91	0,92	0,93	0,94	0,95
Avec mesures de maintenance	0,75384085	0,75319993	0,75333406	0,75115872	0,74408241	0,73970065	0,73884658	0,74013529
Sans mesures de maintenance	0,75409023	0,75155961	0,75180648	0,74847622	0,74228782	0,7379821	0,73825685	0,7386878

FIG. *3.7: Comparaison de la précision en utilisant la base de test.*

Pour tester l'amélioration des résultats fournis par le système en présentant les mêmes cas plusieurs fois, nous avons effectué plusieurs passages de la base d'apprentissage dans le système. Ces tests comparent les résultats obtenus lors du dernier passage de la base d'apprentissage dans le système avec ceux de l'approche qui n'utilise pas les mesures de maintenance, en faisant varier le seuil de similarité. Des améliorations ont été observées dans les résultats (le rappel et la précision) fournis par le système (figures 3.4, 3.5).

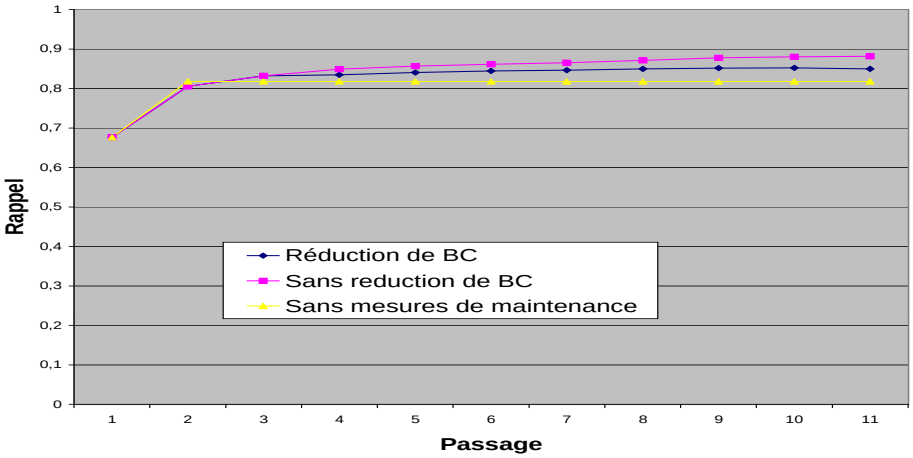


FIG. *3.8: Effet de la réduction de la base de cas sur le rappel.*

Nous avons ensuite effectué des tests en utilisant la base de test (un seul passage de la base dans le système) et nous avons remarqué que les résultats fournis sont légèrement meilleurs en utilisant les mesures de maintenance (figures 3.6, 3.7).

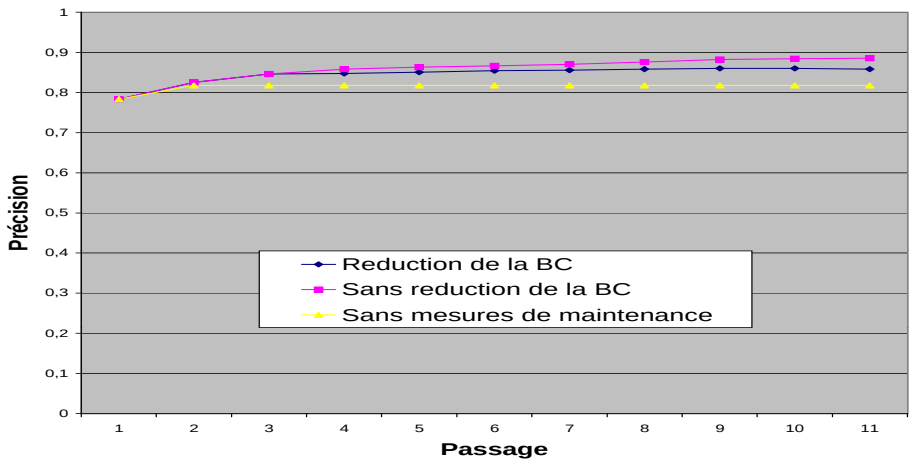
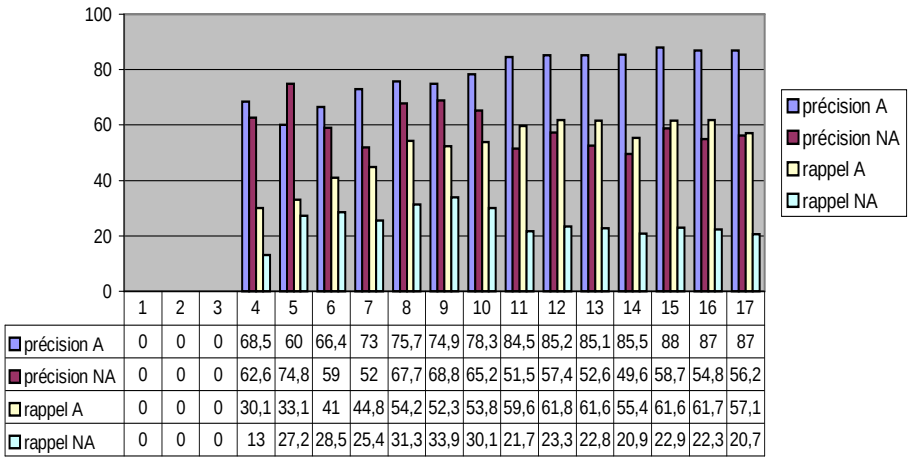


FIG. *3.9: Effet de la réduction de la base de cas sur la précision.*

Dans les tests décrits ci-dessus, la réduction de la base de cas n'a pas été effectuée. Nous avons alors testé l'effet de la réduction de la base de cas, en effectuant plusieurs passages de la base d'apprentissage dans le système, et comparé les résultats obtenus avec ceux obtenus précédemment (figures 3.8, 3.9).



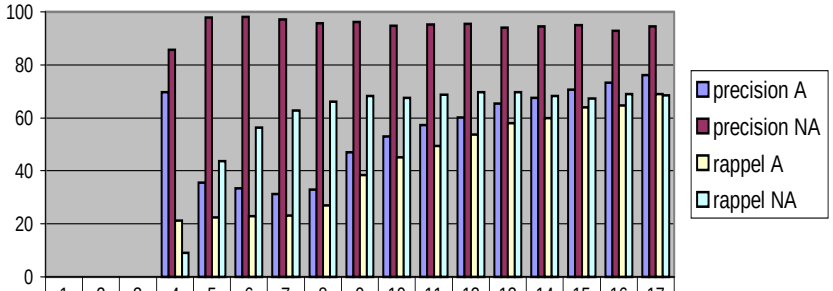
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
précision A	0	0	0	68,5	60	66,4	73	75,7	74,9	78,3	84,5	85,2	85,1	85,5	88	87	87
précision NA	0	0	0	62,6	74,8	59	52	67,7	68,8	65,2	51,5	57,4	52,6	49,6	58,7	54,8	56,2
rappel A	0	0	0	30,1	33,1	41	44,8	54,2	52,3	53,8	59,6	61,8	61,6	55,4	61,6	61,7	57,1
rappel NA	0	0	0	13	27,2	28,5	25,4	31,3	33,9	30,1	21,7	23,3	22,8	20,9	22,9	22,3	20,7

FIG. *3.10: Résultats obtenus sans l'utilisation de la base de cas pour chaque état.*

Nous avons remarqué une légère diminution des résultats en effectuant la réduction (la nouvelle base de cas = 85 % *ancienne base de cas). Mais les résultats obtenus sont toujours meilleurs que ceux fournis par le système sans utilisation des mesures de maintenance.

3.5.2 Validation du système CASEP

Nous avons effectué des expérimentations sur des fichiers de traces de navigations (fichiers log) contenus dans un site Web de commerce électronique, où environ 3000 navigations sont enregistrées chaque jour (voir la description de l'application au paragraphe §1.1). Un ensemble de caractéristiques pour chaque état d'une séquence a été extrait à partir de ces données. Ces caractéristiques ont été obtenues par un pré-traitement des données en utilisant une carte topologique SOM (ce pré-traitement est décrit en détail dans le paragraphe §1.2).
Nous avons conçu et mis en œuvre un système de RàPC afin de prédire, le plus tôt possible, la classe {acheteur, non acheteur} d'un utilisateur du site, en utilisant l'expérience acquise par des internautes ayant navigué de façon similaire. Les séquences représentent les navigations des utilisateurs dans le site. Elles sont enregistrées et stockées dans la base de séquences. Ces séquences contiennent plus de 92 % de non acheteurs et moins de 8 % d'acheteurs et leurs longueurs varient entre 3 et 17 états. Un cas cible représente la succession des 3 derniers états de la navigation courante ($p = 3$). Un état représente un ensemble de pages successives similaires. Cette similarité entre les pages est obtenue par le codage de données décrit dans le paragraphe §1.2 : deux pages sont similaires si elle appartiennent au même neurone dans le codage effectué par la carte de Kohonen. Le classement des internautes {acheteur, non acheteur} est effectué.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
precision A	0	0	0	69,7	35,6	33,3	31,3	32,9	47	53	57,3	60,2	65,4	67,6	70,8	73,2	76,2
precision NA	0	0	0	85,6	97,8	98,1	97,2	95,7	96,1	94,7	95,1	95,6	94,1	94,5	94,9	92,9	94,4
rappel A	0	0	0	21,3	22,3	23	23,2	27	38,5	45,1	49,5	53,8	58,1	59,9	64	64,6	69,1
rappel NA	0	0	0	9,04	43,6	56,4	62,7	66,1	68,3	67,5	68,7	69,7	69,6	68,3	67,3	68,9	68,5

FIG. *3.11: Résultats obtenus avec l'utilisation de la base de cas à chaque présentation d'un état.*

Paramètre	p	e	α_0	d'	δ	A	B	pr
Valeur	3	2	0.7	0.3	0.6	0.3	0.7	0.8

TAB. *3.1: Valeurs des paramètres utilisés dans le système.*

Les valeurs des différents paramètres cités dans la description du système CASEP sont données dans le tableau 3.1.

Les mesures utilisées pour comparer les deux approches sont :

- *La précision :* représente le taux de bons classements parmi les classements proposés par le système ($\frac{nombre\_bon\_classements}{nombre\_classements}$) ;
- *Le rappel :* représente le taux de bons classements parmi tous les classements que le système devrait effectuer ($\frac{nombre\_bon\_classements}{nombre\_cas\_cibles}$) ;
- *Le taux de classement :* représente le taux de classements fournis par le système parmi les classements qu'il devrait effectuer ($\frac{nombre\_classements}{nombre\_cas\_cibles}$).

Nous avons commencé par expérimenter un prototype du système en utilisant seulement la base de navigations, ensuite nous avons construit une base de cas pour laquelle nous avons appliqué notre stratégie de maintenance.

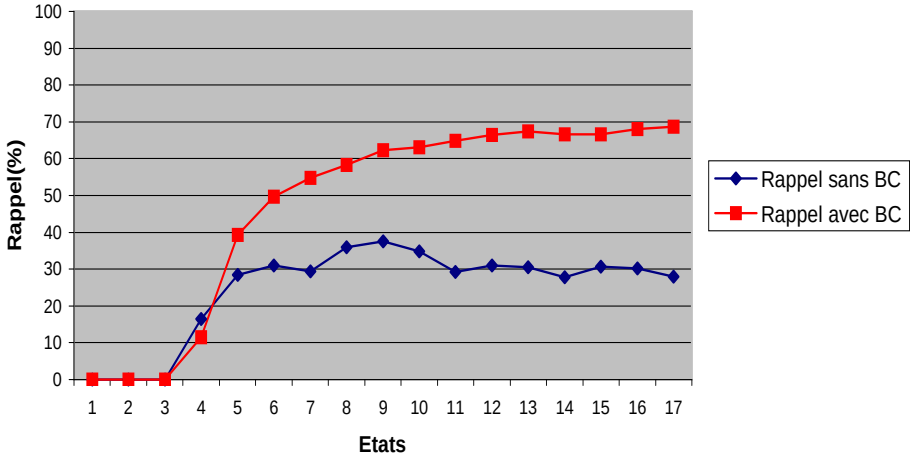


FIG. *3.12: Comparaison du rappel à chaque présentation d'un état.*

Comparaison des résultats obtenus avec et sans base de cas

Quand le système ne contient pas la base de cas dans sa mémoire, nous remarquons que les résultats obtenus pour classer les acheteurs (précision A, rappel A) sont meilleurs que ceux obtenus pour classer les non acheteurs (précision NA, rappel NA) (voir la figure 3.10).

En utilisant la base de cas et les mesures de maintenance, le rappel et la précision sont meilleurs pour le classement des non acheteurs (voir la figure 3.11).

Nous avons effectué des tests (voir les figures 3.12, 3.13 et 3.14) qui comparent les résultats (précision, rappel et taux de prédiction) globaux (sans distinction entre les classes achat et non achat) fournis par le système de RàPC avec et sans la base de cas à chaque état en utilisant 10000 navigations.
Ces tests montrent que les résultats obtenus avec notre système qui contient une base de cas et une stratégie de maintenance (rappel avec BC, précision avec BC, taux de prédiction avec BC) sont meilleurs que ceux obtenus par un système qui utilise seulement la base de navigations (rappel sans BC, précision sans BC, taux de classement sans BC).

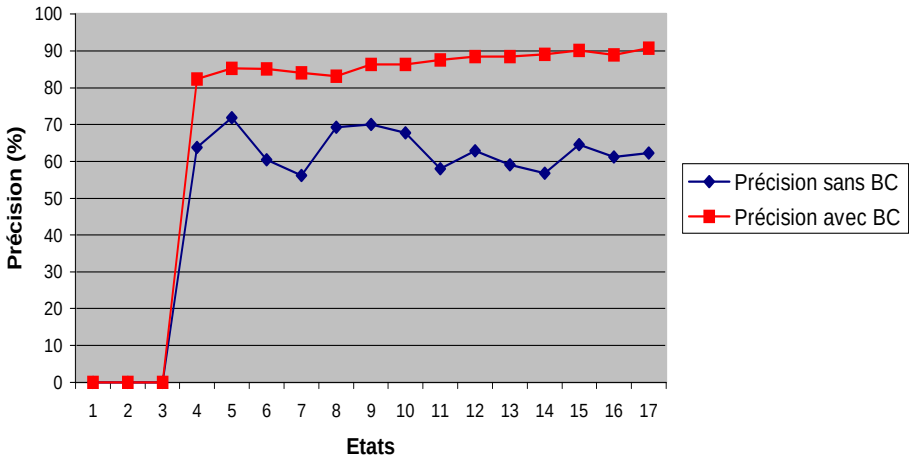


FIG. *3.13: Comparaison de la précision à chaque présentation d'un état.*

Résultats globaux obtenus

Le tableau 3.2 montre les résultats obtenus par le système avec et sans l'utilisation de la base de cas sur tous les états.

Nous remarquons que les résultats obtenus pour classer les non acheteurs s'améliorent avec l'ajout de la base de cas et des mesures de maintenance, tandis que les résultats obtenus pour l'achat sont moins bons. Mais les résultats globaux (rappel, précision et taux de classement)

Résultats globaux.	Sans base de cas	Avec la base de cas
Taux de prédiction	56,09	76,62
Rappel Achat	48,25	43,69
Rappel non Achat	32,44	70,61
Précision Achat	77,46	53,73
Précision non Achat	61,59	95,38
Rappel	38	61
Précision	68	80

TAB. *3.2: Résultats globaux en « achat » et en « non achat ».*

sont nettement meilleurs quand on ajoute la base de cas et les mesures de maintenance car le pourcentage d'acheteur (8 %) est nettement inférieur à celui des non acheteurs (92 %).

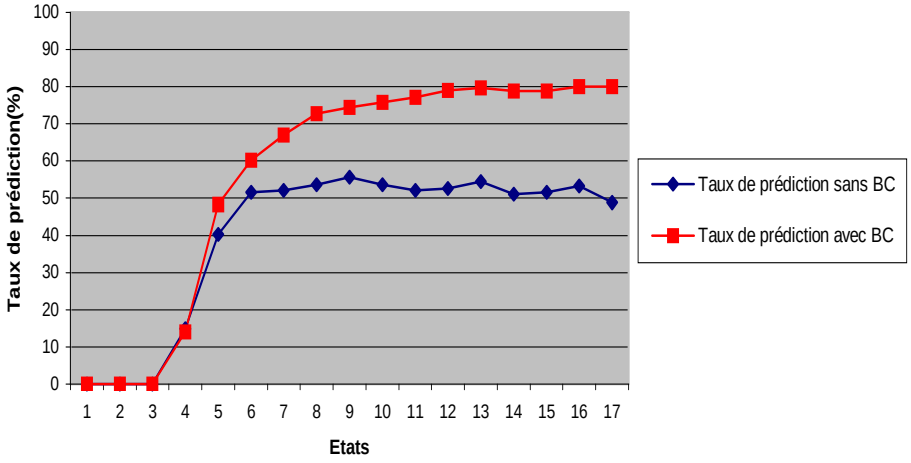


FIG. *3.14: Comparaison du taux de prédiction à chaque présentation d'un état.*

Résultats obtenus en utilisant plusieurs fois la même base

1. ***Deux passages de la même base dans le système***

   Dans ce test, nous avons effectué deux passages d'une même base qui contient 5000 navigations.

   Les résultats (précision A, précision NA, rappel A, rappel NA) fournis à chaque état sont meilleurs quand la base d'apprentissage est passée deux fois comparés à ceux obtenus pré-

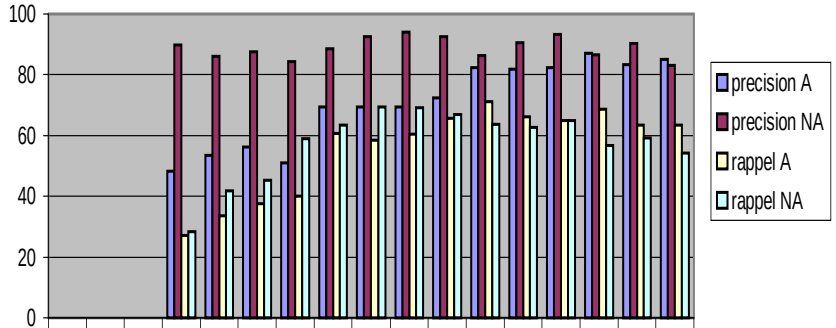
Résultats globaux	
Taux de classement	78,36
Précision non Achat	85,22
Précision Achat	75,23
Rappel Achat	55,50
Rappel Non Achat	68,74
Rappel	63.82
Précision	81,47

TAB. *3.3: Résultats globaux après deux passages de la même base dans le système.*

cédemment (voir les figures 3.15, 3.11).

Résultats globaux

Le tableau 3.3 montre que les résultats globaux obtenus après le deuxième passage de la base d'apprentissage sont meilleurs que ceux obtenus précédemment (voir le tableau 3.2).



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
precision A	0	0	0	48,2	53,5	56,3	50,9	69,3	69,4	69,4	72,4	82,4	81,8	82,3	87,1	83,3	85,1
precision NA	0	0	0	89,8	86,1	87,5	84,3	88,6	92,6	94	92,6	86,4	90,4	93,2	86,6	90,2	83,1
rappel A	0	0	0	27,2	33,5	37,6	40,1	60,7	58,5	60,4	65,7	71,2	66,1	65	68,5	63,4	63,5
rappel NA	0	0	0	28,4	41,7	45,3	59,1	63,3	69,5	69,2	67	63,8	62,8	64,9	56,8	59,2	54,1

FIG. *3.15: Résultats obtenus après deux passages de la même base dans le système.*

Résultats fournis en utilisant la base de cas

Le tableau 3.4 montre la fréquence d'utilisation de la base de cas dans le système (sachant

	Passage1	Passage2
Taux d'utilisation Achat	0,62	0,71
Taux d'utilisation positive Achat	0.581490	0.631988
Taux d'utilisation non Achat	0,6	0,7
Taux d'utilisation positive non Achat	0.780443	0.796361
Taux d'utilisation	0,61	0,7

TAB. *3.4: Résultats fournis par l'utilisation de la base de cas avec deux passages de la même base dans le système.*

Taux de classement	92,26
Précision non Achat	79,56
Précision Achat	89,30
Rappel Achat	70,76
Rappel Non Achat	82,11
Rappel	77,91
Précision	85,28

TAB. *3.5: Résultats globaux après plusieurs passages de la même base dans le système.*

que la base de séquences est aussi utilisée pour le classement) pour effectuer des classements ainsi que les résultats fournis par l'utilisation de celle-ci lors des deux passages.

Le tableau 3.4 montre une évolution positive de la fréquence d'utilisation de la base de cas ainsi que des résultats fournis par le système CASEP en l'utilisant.

2. ***Plusieurs passages de la même base dans le système***
Dans cette expérimentation, plusieurs passages d'une base contenant 1000 navigations sont effectués.

Résultats globaux
Les résultats présentés dans le tableau 3.5 sont obtenus après le 7^{eme} passage de la même base d'apprentissage.

Nous remarquons que, globalement, les résultats obtenus sont nettement meilleurs que ceux présentés précédemment (voir les tableaux 3.2, 3.3).
Résultats fournis par l'utilisation de la base de cas
Le tableau 3.6 montre la fréquence d'utilisation de la base de cas dans le système ainsi que les résultats fournis par son utilisation lors des sept passages dans le système.

	P1	P2	P3	P4	P5	P6	P7
Taux d'utilisation Achat	0,65	0,7	0,68	0,68	0,70	0,71	0,71
Taux d'utilisation positive Achat	0.65	0.65	0.67	0.68	0.68	0.67	0.67
Taux d'utilisation non Achat	0,58	0,66	0,66	0,65	0,67	0,7	0,7
Taux d'utilisation positive non Achat	0.56	0.77	0.74	0.74	0.80	0.87	0.87
Taux d'utilisation	0,61	0,68	0,67	0,66	0,68	0,71	0,71

TAB. *3.6: Résultats fournis par la base de cas avec plusieurs passages de la même base dans le système.*

Nous remarquons qu'en général la fréquence d'utilisation de la base de cas et les résultats fournis par son utilisation s'améliorent avec plusieurs passages de la base d'apprentissage dans le système.

3.5.3 Interprétation des résultats

Les expérimentations que nous avons effectuées sur une grande base de données, indépendamment de notre application, montrent que les mesures de maintenance que nous avons associées aux cas permettent d'améliorer légèrement les résultats du système. Même en réduisant la base de cas, nous avons de meilleurs résultats que quand les mesures de maintenance ne sont pas associées aux cas. De plus, les résultats du système s'améliorent quand les mêmes cas sont présentés plusieurs fois au système. L'amélioration apportée n'est pas très importante car les base de données utilisées ne contiennent pas autant de « cas-bruit » que celles que nous avons utilisées dans notre application.
Dans les expérimentations effectuées sur notre système de prédiction, nous nous sommes focalisés sur l'amélioration de la qualité de prédiction. Une amélioration des résultats (taux de classement, rappel, précision) fournis par le système avec l'ajout de la base de cas et des mesures de maintenance est constatée. De plus, quand les mêmes cas cibles se présentent plusieurs fois, ces résultats sont encore meilleurs et la base de cas est de plus en plus utilisée de manière positive. Ceci est dû à l'effet des mesures de qualité qui permettent de réduire l'utilisation des cas-bruit qui détériorent les résultats fournis par le système.

3.6 Conclusion

Nous avons conçu et implémenté un système de RàPC (CASEP) pour le classement (ou la prédiction) de séquences. Sa mémoire est constituée d'une base de cas et d'une base de séquences.

Parmi les contributions que nous avons apportées dans ce système nous pouvons citer :

- Les cas ajoutés à la base de cas sont des cas extraits à partir de la base de séquences et non des cas cibles résolus comme c'est les cas dans les systèmes classiques de RàPC. De plus, plusieurs cas peuvent être ajoutés à la base de cas à chaque cycle de RàPC. Le choix de ces cas est fait d'abord en effectuant un regroupement des cas qui ont contribué à la résolution du cas cible avec l'algorithme des matrices de similarité, puis en choisissant des représentants en se basant sur les notions de similarité et de diversité.
- Nous avons associé aux cas de la base de cas de nouvelles mesures de maintenance qui permettent de maintenir la base de cas de notre système afin de faire face aux problèmes issus du grand volume de données et de la présence du bruit en se basant sur la prise en compte des succès et des échecs de l'utilisation des cas de la base de cas pour résoudre les cas cibles.
- Le seuil de similarité est variable, il est propre à chaque cas de la base de cas et dépend des mesures de similarité associées aux cas. Ceci permet de filtrer une partie du bruit contenu dans les données.
- Nous avons proposé une phase de réduction de la base de cas du système CASEP qui est basée sur les mesures de maintenance associées aux cas.

Nous ne nous sommes pas intéressés à d'autres aspects du système qui sont aussi importants pour améliorer les résultats. Parmi ces aspects, nous pouvons citer :

- La représentation du cas :
  - Comme nous l'avons mentionné précédemment, un cas dans notre approche est une succession d'évènements instantanés.
  - Un cas, dans notre approche représente une expérience précise dans une séquence. Mais la partie problème du cas cible représente les p derniers états de la séquence courante où p est fixé. Le fait que la structure du cas soit figée constitue un inconvénient pour notre système. Une structure dynamique de cas (cas de longueurs variables, tolérant des trous entre deux états successifs, etc.) est souhaitable pour un système où les données changent constamment.
- Le fait de permettre une structure dynamique de cas peut nécessiter de définir des méthodes pour extraire les caractéristiques et les états pertinents à partir de chaque séquence.
- Même avec la phase de réduction que nous avons prévue dans le système, la base de cas risque d'être volumineuse. Dans les expérimentations que nous avons effectuées sur la grande base de données, la taille de la base de cas réduite représente 85 % de celle de la base de cas initiale, cette réduction n'est pas toujours suffisante pour avoir une base de cas de taille raisonnable afin de pouvoir satisfaire les contraintes temps réel. Une indexation de cas qui permet de réduire l'espace de recherche dans la base de cas peut améliorer nettement l'efficacité du système

- Pour réduire la taille de la base de séquences, nous avons proposé une heuristique simple qui consiste à supprimer les plus anciennes navigations. Ceci permet, certes, de réduire la taille la base de séquences mais ne garantit pas que les séquences retenues soient les plus représentatives de la base de séquences entière.
- La phase de réutilisation de notre système consiste à calculer une confiance associée à chaque solution. Cette confiance dépend de la représentativité des cas et de leur qualité. Le calcul de la confiance se fait en effectuant des moyennes pondérées. Ceci prend en compte les éléments que nous avons jugés pertinents pour le calcul de la confiance, mais d'autres méthodes plus élaborées peuvent permettre de construire la solution du cas cible.

Pour améliorer notre système « CASEP », nous avons construit un nouveau système baptisé « CASEP2 » où :

- La structure de cas cible est plus générale : Les cas cibles peuvent être de différentes longueurs ;
- Une réduction de l'espace de recherche est effectuée par partition de la base de cas ;
- La phase de réutilisation est améliorée.

Nous nous sommes intéressés particulièrement à construire un système hybride qui combine le RàPC avec les RN. Ceux-ci sont connus pour leur capacité à traiter de grands volumes de données et pour leurs efficacité à effectuer ce traitement, ils sont utilisés dans le système pour :

- indexer les cas de la base de cas du module de RàPC et fournir des solutions à certains cas cibles ;
- effectuer la phase de réutilisation du cycle de RàPC.

Pour effectuer l'hybridation du raisonnement à partir de cas avec un réseau connexionniste, celui-ci doit vérifier certaines propriétés. D'abord, le réseau connexionniste doit être capable d'effectuer une classification (pour l'indexation de la base de cas) et un classement (pour fournir des solutions dans la phase de réutilisation) de séquences. De plus, il doit être capable de faire face à l'arrivée continue des données tout en préservant les connaissances acquises par les apprentissages précédents (propriétés de plasticité et de stabilité). Pour ceci, nous proposons dans ce qui suit de nouveaux modèles de cartes auto-organisatrices qui satisfont ces propriétés.

Deuxième partie

Utilisation des cartes auto-organisatrices

Chapitre 4

Systèmes d'apprentissage connexionnistes

Dans ce chapitre, nous présentons les différentes manières d'insérer l'information temporelle dans les réseaux connexionnistes et particulièrement dans les cartes auto-organisatrices. Nous décrivons, ensuite, certains modèles de cartes évolutives qui prennent en compte l'arrivée continue des données puis les modèles basés sur la théorie de résonance adaptative qui, en plus, sont capables de préserver les connaissances déjà acquises.

4.1 Réseaux connexionnistes et traitement des séquences temporelles

L'utilisation des réseaux de neurones « classiques » comme par exemple, le perceptron multicouches (MLP) (Rosenblatt, 1958) et (Milgram, 1993) ou les cartes auto-organisatrices de Kohonen (Kohonen, 1995), est destinée au traitement des données statiques. Ces modèles classiques ne sont pas conçus pour traiter des données qui varient avec le temps : « les séquences temporelles ».
Pour traiter les séquences temporelles, des modèles connexionnistes temporels adaptés sont utilisés. Ceux-ci constituent un domaine de recherche en pleine évolution.
Plusieurs aspects du temps peuvent être utilisés dans les problèmes temporels. Ces aspects correspondent aux différentes utilisations du temps (Chappelier *et al.*, 2000) comme :

- Le temps est représenté par une simple relation d'ordre (le temps est un index utilisé pour ordonner les évènements) ;
- Le temps est considéré comme une mesure (par exemple, le temps considéré dans le traitement de la parole où la durée est significative) ;
- le temps est discret ou continu ;

– Le temps est représenté dans un intervalle fini ou infini (des problèmes qui ne se terminent jamais).

L'ordre chronologique des modèles temporels connexionnistes, proposés dans la littérature, est caractérisé par l'augmentation de l'intégration du temps dans leurs architectures.
Les premiers réseaux connexionnistes temporels sont caractérisés par des architectures basés sur des modèles classiques, mais modifiés localement pour tenir compte de la dimension du temps. Par exemple, les réseaux de neurones récurrents (Jordan, 1986) et (Rumelhart *et al.*, 1986) basés sur les MLP ont été introduits. Ils contiennent des liens de rétroaction « *backward* » avec un délai d'une étape temporelle qui représente la relation d'ordre entre deux entrées successives.
La seconde phase considère toujours les architectures classiques, elle s'est focalisée sur la configuration de leurs paramètres de façon globale. Le vecteur d'entrée est l'un des paramètres qui peuvent prendre en compte la dimension du temps (par exemple, il peut représenter une fenêtre temporelle dans le signal d'entrée « *Time Delay Neural Network* : TDNN » (Waibel *et al.*, 1989)).
Après l'utilisation des architectures classiques adaptées, de nouvelle architectures conçues pour le traitement des données temporelles ont été développées. La plupart de ces architectures sont inspirées des connaissances sur le traitement d'informations dans les réseaux de neurones naturels (Euliano, 1998). Elles essayent d'imiter une ou plusieurs caractéristiques comme :

– Le délai de propagation sur les connexions ;
– Les neurones qui envoient des activités discrètes ou continues ;
– L'activité du neurone qui peut être une fonction du temps.

Ceci ne veut pas dire que les autres modèles ont été abandonnés puisque de nouvelles extensions des modèles récurrents (Baldi *et al.*, 1999) et (Hochreiter et Schmidhuber, 1996) et de nouveaux modèles avec représentation externe du temps (Kohonen, 1992) et (Somervuo et Kohonen, 1999) ont été aussi proposés dans la littérature.

4.2 Intégration du temps dans les modèles connexionnistes

Dans ce paragraphe, nous allons décrire les différentes approches de l'intégration du temps dans les modèles connexionnistes cités précédemment. Une hiérarchie des modèles temporels a été proposée dans (Chappelier *et al.*, 2000) (voir la figure 4.1).

1. **Représentation externe du temps**
   L'intégration du temps dans ces modèles connexionnistes consiste à le représenter en dehors du réseau de neurones.
   Un pré-traitement des données est effectué de façon à ce que les modèles connexionnistes classiques puissent prendre en compte l'aspect temporel des données. Le temps est alors transformé en une information dans l'espace d'entrée ; le réseau va seulement traiter les

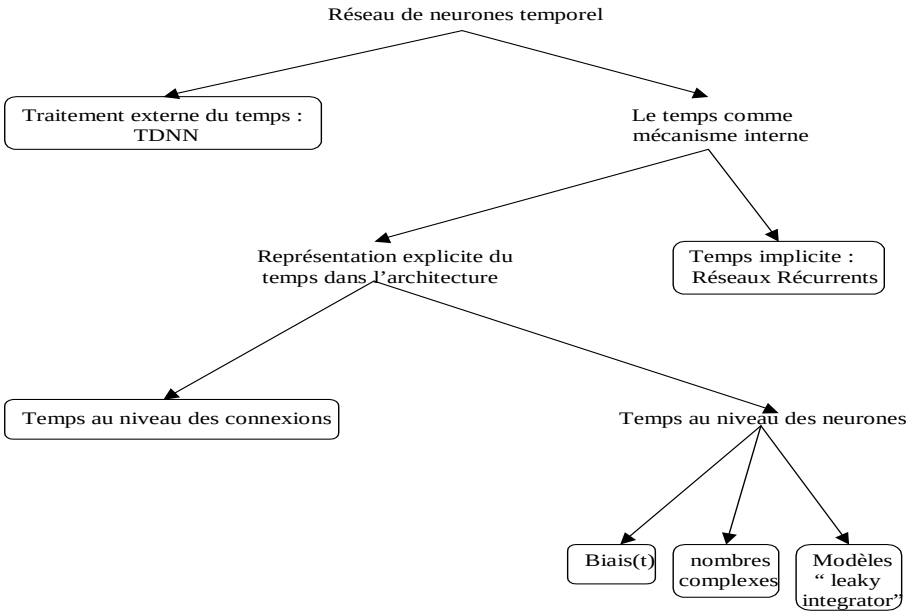


FIG. *4.1: Une classification des modèles connexionnistes selon l'intégration du temps.*

informations de l'espace d'entrée.
Plusieurs modèles utilisant cette approche ont été proposés dans la littérature (Waibel *et al.*, 1989), ils seront détaillés dans le paragraphe §4.2.3.

2. **Le temps comme un index interne**
Le temps peut être introduit dans les modèles connexionnistes à différents niveaux.

– ***Représentation implicite du temps***
Le temps peut être utilisé comme un index dans une séquence d'états du réseau. Il n'y a pas une vraie représentation du temps dans le réseau, mais plus une utilisation du temps comme variable qui contrôle le mécanisme interne du réseau. Dans ce cas, on dit que le temps est présent *implicitement* dans le modèle. Cette sorte de réseau est illustrée par les réseaux de neurones récurrents (Elman, 1990), (Jordan, 1986), (Rumelhart *et al.*, 1986), (Baldi *et al.*, 1999) et (Hochreiter et Schmidhuber, 1996).

– ***Représentation explicite du temps***
Une façon d'introduire le temps dans les modèles neuronaux est de le représenter explicitement au niveau du réseau, soit au niveau des connexions ou bien au niveau des neurones (ou les deux).

– *Le temps au niveau des connexions*
Ceci est habituellement réalisé en utilisant des délais de propagation sur les connexions

« poids temporels ».

- *Le temps au niveau des neurones*
  L'introduction du temps au niveau du neurone peut être effectuée en simulant des propriétés biologiques ou bien en introduisant le temps sans inspiration biologique (en ingénierie) (Chappelier *et al.*, 2000).
  La première approche mène habituellement à des modèles neuronaux basés sur des équations différentielles. Le modèle le plus utilisé dans ce contexte est le « *leaky integrator* » (Barreto et Arajo, 2001). Le principe sous-jacent à ces modèles consiste à sommer les entrées du neurone sur une période de temps. Quand cette somme devient supérieure à un certain seuil (spécifique pour chaque neurone), le neurone change d'état.
  L'approche d'ingénierie est purement algébrique, elle consiste à effectuer l'une des opérations suivantes :

  - Changer la représentation habituelle (scalaire) en des nombres complexes ;
  - Introduire un biais variable du temps artificiel ;
  - Généraliser l'équation standard des réseaux de neurones, ce qui rend le modèle obtenu équivalent au « *leaky integrator* ».

D'après Elman (Elman, 1990), l'utilisation d'une fenêtre temporelle pour prendre en compte le temps (représentation externe) entraîne plusieurs inconvénients : elle impose des limites rigides sur la durée de l'entrée, ce qui n'est pas nécessairement approprié aux entrées temporelles dans les applications réelles. En plus, elle souffre du problème causé par les fenêtres temporelles non adéquates et non flexibles et du sur-apprentissage provenant d'un nombre excessif de poids.
Les réseaux de neurones temporels inspirés du fonctionnement des réseaux de neurones naturels paraissent comme étant les plus adaptés pour traiter les séquences temporelles, mais les recherches dans ce domaine sont récentes et des études approfondies sur les différentes approches existantes et leur efficacité dans la résolution de différents problèmes sont nécessaires. Les modèles effectuant une représentation externe de l'information temporelle peuvent aussi être efficaces dans certaines applications (Kangas, 1991) et (Kohonen, 1992).

4.3 Extensions temporelles des cartes topologiques de Kohonen

Plusieurs efforts ont été effectués pour introduire l'information temporelle dans la carte de Kohonen (Kohonen, 1995). Ces modèles permettent de prendre en compte les informations spatiales et temporelles des données en utilisant différentes techniques (voir le paragraphe §4.1).
Comme dans les autres réseaux connexionnistes (Barreto et Arajo, 2001) :

- la première technique modifie localement la carte SOM classique (cartes topologiques récurrentes)
- la deuxième technique consiste à ajouter cette information à l'extérieur de la carte topologique
- la troisième introduit cette information à l'intérieur de la carte.

4.3.1 Les cartes topologiques récurrentes

Dans ces modèles, des modifications locales sont introduites dans la carte SOM pour tenir compte de l'aspect temporel des données.

4.3.1.1 Carte auto-organisatrice récurrente (RSOM)

Dans la carte auto-organisatrise (RSOM : Recurrent Self Organizing Map) (Varsta *et al.*, 1997), l'entrée $x(t)$ à l'instant t est modélisée suivant l'équation :

$$y_i(t,\alpha) = (1-\alpha)y_i(t-1,\alpha) + \alpha(x(t) - w_i(t)) \tag{4.1}$$

où $y_i(t,\alpha)$ est un vecteur appelé « vecteur différence (*leaked difference*) » du neurone c_i à l'instant t, le coefficient α $(0 < \alpha < 1)$ « *leaking coefficient* » représente la profondeur de la mémoire, $w_i(t)$ est le vecteur poids du neurone c_i à l'étape t et $y_i(t,0) = 0$.

Après la transformation de l'entrée en un vecteur différence, la carte est traitée comme une carte SOM normale. Comme dans SOM, le neurone gagnant est défini par :

$$s(t) = \arg\min_{c_i \in A} ||y_i(t,\alpha)|| \tag{4.2}$$

où A est l'ensemble des neurones.

Après le choix du neurone gagnant, une fonction d'erreur est définie dans le but de minimiser pour l'entrée $x(t)$ le vecteur différence $y(t,\alpha)$ comme suit :

$$E(t) = \frac{1}{2}\sum_{c_i \in A} h(s(t),i)y_i(t,\alpha)^2 \tag{4.3}$$

où h(,) est une fonction de voisinage.

La règle d'adaptation stochastique qui minimise l'erreur $E(t)$ est donnée par :

$$w_i(t+1) = w_i(t) + \epsilon(t)h(s(t),i)y_i(t,\alpha) \tag{4.4}$$

où $\epsilon(t)$ est le pas d'apprentissage et h(,) est une fonction de voisinage.

4.3.1.2 Carte auto-organisatrice récurrente pour le traitement des séquences temporelles

Ce modèle (*Recurrent self organizing map for temporal sequence processing*) (McQueen *et al.*, 2002) effectue la classification de chaque entrée en se basant sur le vecteur d'entrée et sur un vecteur contexte. Celui-ci fournit une représentation à deux dimensions du neurone gagnant précédent. Cette récurrence permet au réseau de traiter les séquences temporelles. Ce modèle a été appliqués aux problèmes difficiles de traitement de la langue naturelle.
Les entrées du réseau sont des vecteurs de 28 bits qui fournissent une représentation binaire de l'entrée traitée. En plus de ce vecteur d'entrée, le réseau utilise un vecteur contexte. La taille de ce vecteur contexte peut varier selon la taille du réseau (une taille de 10 bits est utilisée dans les expérimentations). Les deux vecteurs d'entrée et de contexte sont utilisés dans le calcul de la distance euclidienne pour déterminer le neurone gagnant de façon similaire à SOM.
Le vecteur contexte représente le neurone gagnant précédent en utilisant un vecteur de 10 bits. Les cinq bits de gauche représentent le numéro (binaire) de la colonne gagnante, tandis que les cinq bits de droite représentent le numéro (binaire) de la ligne du neurone gagnant. Cette approche représente une amélioration d'une approche initiale qui n'utilise que la représentation binaire des entrées de la carte SOM. Comme dans SOM, une fonction de voisinage est utilisée pour mettre à jour les poids des neurones dans la région autour du neurone gagnant.

4.3.2 Les cartes topologiques à mémoire externe

Ces cartes effectuent un codage externe des entrées. Si une séquence finie $X(t)$ de n vecteurs est présentée comme une entrée de SOM, chaque neurone de la carte peut correspondre à un opérateur qui l'analyse.

4.3.2.1 Version temporelle de SOM avec délai exponentiel pondéré

Kangas (Kangas, 1991) a proposé une variante temporelle de SOM pour représenter les caractéristiques séquentielles des données d'entrée. Ces modèles ont été beaucoup appliqués pour des tâches de reconnaissance de phonèmes. Dans le modèle de la figure 4.2, l'entrée $x'(t)$ de SOM est une fonction des entrées précédentes, elle est donnée par :

$$x'(t) = \lambda x(t) + (1 - \lambda)x'(t - 1) \tag{4.5}$$

où $x(t)$ est le vecteur de la séquence courante, $0 \leq \lambda \leq 1$ est le paramètre de la profondeur de la mémoire qui détermine l'influence des entrées passées et $x'(0) = x(0)$.

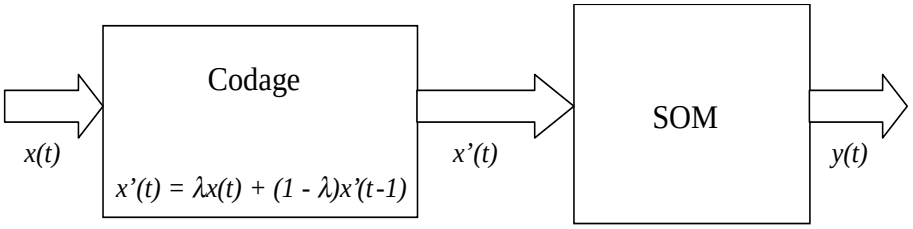


FIG. *4.2: Version temporelle de SOM avec un décalage exponentiel pondéré.*

4.3.2.2 Version temporelle de SOM

Le modèle illustré dans la figure 4.3 représente les séquences par des successions de fenêtres temporelles (Kangas, 1991) : les T derniers composants de la séquence d'entrée sont concaténés et présentés au réseau. Cette procédure nécessite beaucoup de calculs et augmente considérablement le temps d'apprentissage. Cependant, cette stratégie produit de bons résultats dans les tâches de reconnaissance (Kangas, 1991).

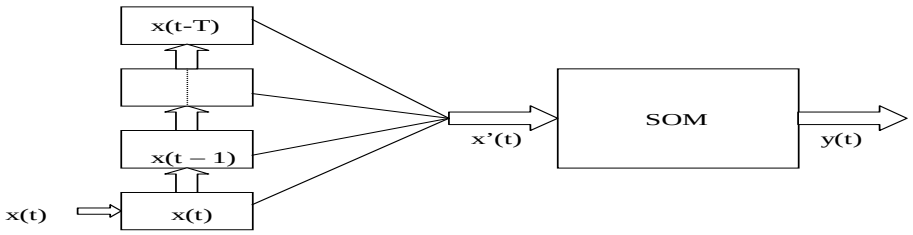


FIG. *4.3: Version temporelle de SOM.*

4.3.2.3 Architecture Hypermap

Dans ce modèle (Kohonen, 1992), différentes fenêtres temporelles centrées sur une étape de temps particulière t sont utilisées pour construire deux types de vecteurs d'entrée du réseau (figure 4.4). L'idée de l'Hypermap est fondée sur le fait qu'une forme dans une séquence est généralement contenue dans le contexte d'une autre forme. Donc deux niveaux d'informations d'entrée sont utilisés pour la sélection du gagnant.
D'abord, un vecteur patron $x_{pat}(t)$ est formé en effectuant la concaténation de plusieurs composants de la séquence autour de l'instant t. Puis un plus grand vecteur $x_{cont}(t)$, considéré comme le contexte de $x_{pat}(t)$ est construit. Ces vecteurs sont ensuite présentés à la carte SOM où chaque neurone possède deux groupes d'entrées, un pour $x_{pat}(t)$ et un autre pour $x_{cont}(t)$. Le vecteur contexte est d'abord utilisé pour sélectionner un sous-ensemble de neurones de la carte. Dans ce sous-ensemble, le neurone le plus proche est choisi en utilisant le vecteur patron. Cette approche a été utilisée avec succès pour les problèmes de reconnaissance, de traitement des séquences bio-

logiques et de la reconnaissance de la parole (Kohonen, 1992) .

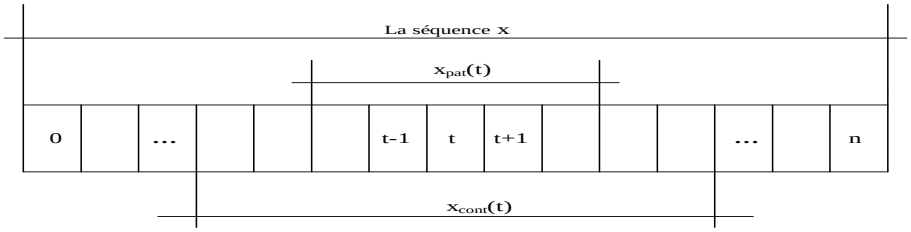


FIG. *4.4: Architecture Hypermap.*

4.3.2.4 Carte auto-organisatrice d'unités auto-régressives

Les états futurs d'un processus auto-régressif (AR) sont définis par ses états précédents et une composante de bruit aléatoire. Dans le cas linéaire, ce processus est représenté par une somme pondérée des valeurs précédentes du processus plus le bruit. Un modèle AR linéaire pour un processus discret x(t) est donné par :

$$x(t) = \sum_{k=1}^{N} m(k)x(t-k) + \gamma(t), \quad t \geq N \tag{4.6}$$

où N est le degré du modèle et $\epsilon(t)$ représente le bruit. Les N vecteurs de coefficients $m(k)$ $(k = 1, ..., N)$ peuvent être obtenus, par exemple, avec la méthode des moindres carrés.

Dans la carte auto-organisatrice de modèles auto-régressifs (AR-SOM) (Lampinen et Oja, 1989), les vecteurs poids des neurones sont les coefficients des modèles AR correspondant et le modèle de bruit est supposé Gaussien.
L'apprentissage stochastique de AR-SOM est d'abord effectué par la sélection du neurone gagnant :

$$s(t) = \arg\min_{c_i \in A}[x(t) - \sum_{k=1}^{N} w_i(k)x(t-k)] \tag{4.7}$$

où A est l'ensemble des neurones et w_i est le vecteur poids du neurone c_i.
La mise à jour des coefficients AR du neurone gagnant et de son voisinage est donnée par :

$$w_i(t) = w_i(t-1) + \epsilon_{s(t),i}\gamma_i(t)x(t) \tag{4.8}$$

où x(t) est le vecteur qui représente les N vecteurs précédents et $\gamma(t)$ est l'erreur estimée et ϵ est le pas d'apprentissage.

4.3.2.5 SOM avec une distorsion temporelle dynamique

La programmation dynamique (Bellman, 1957) permet le calcul des distances entre les mots de l'alphabet en calculant le coût minimum de translation d'un mot à un autre. Un coût est associé à chaque opération possible comme la suppression d'une lettre, l'insertion d'une lettre ou le remplacement d'une lettre par une autre dans l'alphabet.
La DTW (*Dynamic Time Warping*) (Boyer *et al.*, 1987) est un algorithme de programmation dynamique, où une séquence d'observations est souvent comparée à une autre séquence référence. L'association de telles références aux neurones de SOM et l'utilisation de la DTW dans la phase de compétition a donné lieu au modèle DTW-SOM (Somervuo et Kohonen, 1999).
Dans la DTW, les séquences de vecteurs caractéristiques sont comparées en utilisant la fonction de distorsion qui permet de les aligner. Le chemin de distorsion P peut être décrit par une matrice dense de taille $m \times n$, où m et n sont les longueurs des séquences comparées. Chaque élément p_{ij} de la matrice donne l'influence de la distance $d(x_i, y_j)$ entre le i^{eme} élément x_i de la première séquence X et le j^{eme} élément y_j de la séquence référence Y sur la distance globale dans le chemin de distorsion.
Le chemin de distorsion minimal entre les deux séquences X et Y est donné par :

$$D_{min}(X,Y) = \sum_{P_{min}} p_{i,j} d(x(i), y(j)) = \min_{P} \sum_{P} P_{n,m} d(x(i), y(j)) \tag{4.9}$$

où d(.,.) est une distance dépendante de l'application.

4.3.3 Les cartes topologiques à mémoire interne

Dans ces cartes, l'information temporelle est introduite à l'intérieur de la carte topologique.

4.3.3.1 Carte temporelle de Kohonen

La carte temporelle de Kohonen (TKM : *Temporal Kohonen Map*) (Chappell et Taylor, 1993) maintient l'historique de l'activité de chaque neurone c_i en utilisant une variable $a_i(t, \lambda)$ définie comme suit :

$$a_i(t,\lambda) = \lambda a_i(t-1,\lambda) - \frac{1}{2} dist(x(t), w_i(t))^2 \tag{4.10}$$

où $0 \leq \lambda \leq 1$ est la constante de profondeur de la mémoire, *dist* est une fonction de distance, $x(t)$ est le vecteur d'entrée, $w_i(t)$ est le vecteur de poids du neurone c_i et $a_i(0,\lambda) = 0$.
Un neurone de sortie dans TKM stocke l'activité au temps t. Cette activité est ensuite diminuée

d'un pourcentage donné par la constante λ.
Après t étapes de temps, l'activation peut s'écrire comme suit :

$$a_i(t,\lambda) = -(1/2)\sum_{k=0}^{t-1}\lambda^k dist(x(t-k), w_i(t-k))^2 + \lambda^t a_j(0,\lambda) \tag{4.11}$$

Le neurone gagnant c_s choisi est celui qui maximise l'activation :

$$a_s(t,\lambda) = \max_{c_i \in A} a_i(t,\lambda) \tag{4.12}$$

Ses poids et ceux de ses voisins sont mis à jour comme dans SOM. Cette stratégie a été utilisée avec succès dans le classement d'un mot ayant la même position dans un ensemble de phrases de contextes différents (Chappell et Taylor, 1993).

4.3.3.2 Modèle SARDNET

James et Miikkulainen (James et Miikkulainen, 1995) ont introduit une version simple de la carte SOM pour prendre en compte l'aspect temporel des données. Le modèle SARDNET (*Sequential Activation Retention and Decay NETwork*) inclut un mécanisme simple de mémorisation et d'affaiblissement dans le but de former un ensemble unique de neurones actifs pour chaque séquence d'entrée. Deux étapes supplémentaires sont ajoutées à la phase d'apprentissage de la carte SOM.

- La première étape affecte la valeur 1 à l'activation de l'unité gagnante et l'exclut des compétitions suivantes.
- La deuxième met à jour les activations $a_i(t)$ de toutes les unités de la carte de la manière suivante :

$$a_i(t+1) = \xi a_i(t) \tag{4.13}$$

où $0 < \xi < 1$ est un paramètre d'affaiblissement.

A la fin de la présentation de la séquence, le neurone avec la plus grande activation représente le vecteur d'entrée le plus récent et celui avec la plus petite activation représente le premier vecteur de la séquence d'entrée.

4.3.3.3 Modèle basé sur des regroupements neuronaux

Le modèle proposé dans (Durand et Alexandre, 1995) est inspiré des regroupements neuronaux de la colonne corticale. Une nouvelle unité fonctionnelle de base, constituée d'un groupe de neurones, a été conçue pour prendre en compte les informations temporelles des données. La

carte temporelle contient trois sortes de connexions :

- Les poids *feed-forward* qui correspondent à l'information spatiale. L'apprentissage de ces poids est auto-organisé.
- Les connexions intra-carte (intra-connexions) qui définissent le champs de réception temporel de l'unité.
- Les connexions inter-carte (inter-connexions) qui sont seulement utilisées pour l'étiquetage des unités.

Dans la figure 4.5, un exemple de la connexion entre unités (petits cercles) est montré. De telles unités sont incluses dans la carte afin de mémoriser et de remémorer des séquences extraites de l'espace d'entrée. La propagation de l'activité à travers les intra-connexions caractérise la dynamique de la forme à reconnaître. La séquence d'activités correspond à la séquence d'événements dans la forme. Pour éviter qu'une unité soit partagée par plusieurs séquences, une super unité est conçue. Elle permet de rassembler les unités qui réagissent différemment au même stimulus à cause de leurs connexions *feed-forward*.

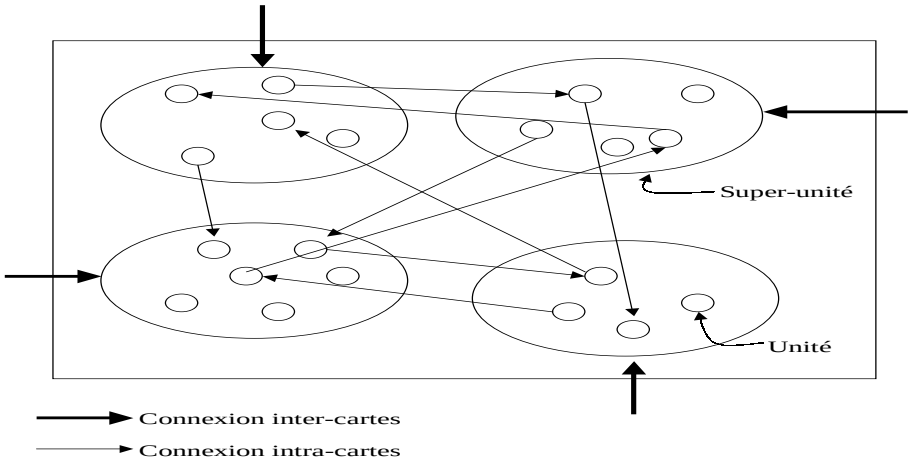


FIG. *4.5: Une carte spatio-temporelle*

L'activation $a_{i,k}(t)$ de l'unité U_i dans une super unité SU_k est donnée par la formule suivante :

$$a_{i,k}(t) = \begin{cases} \bar{a}_{i,k}(t) \ si \ \bar{a}_{i,k}(t) \geq A \\ 0.9 * a_{i,k}(t-1) \ sinon \end{cases} \tag{4.14}$$

où : $\bar{a}_{i,k} = \frac{1}{\alpha+\beta}(\alpha A_S + \beta A_T)$, A est un seuil proche de 1, A_S est l'activité spatiale, A_T est l'activité temporelle.

Le processus d'apprentissage utilise une carte de Kohonen (Kohonen, 1995) pour extraire l'information temporelle et auto-organiser les super unités de la carte. Ce processus est effectué avant l'apprentissage dynamique des intra-connexions. Le processus d'apprentissage est pseudo auto-organisé car pour chaque forme à apprendre, la dernière unité de la séquence est connectée à un niveau plus haut de super-unités correspondant à la forme.

4.3.3.4 Carte temporelle pour la reconnaissance des formes

Ce modèle (SOTPAR : *Self Organizing Temporal Pattern Recognizer*) est basé sur la diffusion de l'activité à travers le temps dans les neurones de la carte topologique de Kohonen et sur l'affaiblissement temporel de ces activités (Euliano et Principe, 1996) et (Euliano, 1998). Le mécanisme de diffusion d'activité permet de garder l'information temporelle dans le réseau. Quand un neurone est activé, son activité est diffusée à ses voisins leurs permettant d'avoir une plus grande probabilité d'être activés après. Cette diffusion d'activité a été inspirée de la diffusion du gaz d'oxyde nitrique qui peut agir comme une trace de mémoire dans le cerveau et qui convertit les corrélations temporelles des entrées en poids de connexions spatiales.
En résumé, il y a trois concepts fondamentaux dans SOTPAR :

- Apprentissage compétitif avec fonction de voisinage ;
- Diffusion d'activité à travers l'espace des neurones ;
- Affaiblissement temporel de ces activités.

En utilisant ces trois concepts, l'information temporelle est convertie et distribuée en des connexions spatiales et des distributions de neurones dans le réseau, en utilisant les principes d'auto-organisation.
Pour chaque entrée $x_{t,k}$ du réseau de neurones, le neurone gagnant c_s est sélectionné de la manière suivante :

$$c_s = \arg\min_{c_i \in A}[dist(x_{t,k}, w_i) - \beta a_i(t)] \tag{4.15}$$

Où : $a_i(t)$ est l'activité diffusée au neurone c_i à l'instant t et β est une constante spatio-temporelle.
La diffusion de l'activité temporelle s'effectue de la manière suivante :

$$a_i(t) = (1-\alpha)(a_i(t-1) + \delta_{c_i,c_s(t-1)}) + (1-\alpha)(a_{vois(c_i)}(t-1) + \delta_{vois(c_i),c_s(t-1)}) \tag{4.16}$$

où :
$\alpha \leq 1$ est la constante d'affaiblissement appliquée pour affaiblir l'activité, $a_{vois(c_i)}(t)$ est l'activité du voisin de c_i, $vois(c_i)$ représente le voisin du neurone c_i qui se trouve dans la direction de propagation, $a_i(0) = 0$ et δ est la fonction de Kroneker définie par :

$$\delta_{ij} = \begin{cases} 1 \; si \; i = j \\ 0 \; si \; i \neq j \end{cases}$$

Modèle	Sélection du gagnant	But de l'adaptation des poids
TKM	$\max a_i(.,\lambda)$	$\max a_i(.,0)$
RSOM	$\min \|\|y(.,\alpha)\|\|^2$	$\min \|\|y(.,\alpha)\|\|^2$

TAB. *4.1: Comparaison des propriétés de TKM et RSOM.*

L'adaptation des poids est effectuée de la même manière que dans la carte de Kohonen.

D'autres modèles de cartes auto-organisatrices à représentation interne ont été proposés récemment dans la littérature (B.Hammer *et al.*, 2004). Parmi ces modèles, nous pouvons citer :

- la carte RecSOM : Recursive SOM (Voegtlin, 2002) qui inclut les activités de tous les neurones de la carte dans le calcul de l'activité d'un neurone.
- La carte SOMSD : SOM for Structured Data (Hagenbuchner *et al.*, 2003) qui inclut l'index du neurone gagnant précédent dans le calcul de l'activité de chaque neurone de la carte.
- La carte MSOM : Merge SOM (M.Strickert et B.Hammer, 2004) qui inclut le contenu du neurone gagnant précédent dans le calcul de l'activité des neurones.

4.3.4 Discussion

En plus du classement des cartes topologiques temporelles proposé dans la littérature, nous pouvons distinguer les modèles qui utilisent des fenêtres temporelles pour tenir compte de l'aspect temporel des données, comme la version temporelle de SOM (Kangas, 1991) et la carte auto-organisatrice d'unités auto-regressive (Lampinen et Oja, 1989) qui utilisent les derniers composants de la séquence ainsi que l'architecture Hypermap (Kohonen, 1992) qui utilise des vecteurs qui concatènent les composants de la séquence se trouvant autour d'un instant t (ici les derniers composants de la séquence ne sont pas favorisés par rapport aux autres), de ceux utilisant les informations sur la totalité de la séquence. Parmi ces derniers modèles, il y a une différence entre les modèles où chaque instant a la même importance comme dans la carte auto-organisatrice avec une distorsion temporelle dynamique (Somervuo et Kohonen, 1999) , de ceux où les derniers instants d'une trajectoire sont favorisés comme dans la carte auto-organisatrice récurrente (Barreto et Arajo, 2001), la version temporelle de SOM avec décalage exponentiel pondéré (Kangas, 1991), le modèle SARDNET (James et Miikkulainen, 1995) et le modèle SOTPAR (Euliano et Principe, 1996).
Les réseaux RSOM et TKM se ressemblent beaucoup, mais certaines différences existent entre ces deux modèles. Des comparaisons analytique et expérimentales des propriétés de RSOM avec celles de TKM ont été effectuées dans (Varsta *et al.*, 2001). Les règles d'apprentissage de TKM et de RSOM sont résumées dans le tableau 4.1

Une analyse mathématique a été suffisante pour montrer que la maximisation de l'activité dans TKM est similaire à la minimisation du vecteur de différence dans RSOM quand les deux

cartes partagent la même topologie et les mêmes données. Comme la règle d'adaptation de RSOM est une descente de gradient qui minimise la somme des normes quadratiques du vecteur de différence régularisé par le voisinage, la carte cherche explicitement à apprendre les poids définis précédemment. Ce qui n'est pas le cas de TKM : les auteur de (Varsta *et al.*, 2001) ont montré que généralement le poids de l'état d'équilibre de TKM ne maximise pas l'activité et utilisent des simulations pour monter comment ceci affecte le comportement de TKM. Celui-ci concentre la plupart de ses neurones dans les bords et les coins de l'espace des entrées laissant seulement quelques neurones pour couvrir le reste de l'espace d'entrée. Le modèle RSOM par contre permet d'effectuer un apprentissage sur toutes les entrées distribuées dans l'espace de manière équilibrée.
Les modèles récents à représentation interne du temps sont construits selon le même principe que TKM, sauf que le contexte de l'activité temporelle est différent. Dans TKM, seul l'activité précédente du neurone compte pour le calcul de son activité courante, tandis que dans RecSOM, MSOM et SOMSD, d'autres neurones de la carte sont utilisés dans le calcul de cette activité.

D'autres modèles temporels hiérarchiques ont été proposés dans la littérature (Kangas, 1991) et (Privitera et Shastri, 1996). Ces modèles combinent plusieurs cartes auto-organisatrices et permettent d'effectuer des tâches complexes.

Les cartes auto-organisatrices citées possèdent une structure fixée au préalable et ne peuvent pas faire face à l'arrivée continue des données ou effectuer un apprentissage à long terme. Lorsque de nouvelles données arrivent, la carte obtenue en utilisant une ancienne base de données ne peut être utilisée pour apprendre les nouvelles. Le nombre de neurones peut ne pas être adéquat et l'apprentissage effectué sur les anciennes données sera perdu. Il y une autre alternative qui consiste à construire une nouvelle carte en utilisant toutes les données, ceci est très coûteux en temps de calcul et même en mémoire et ne peut être envisagé pour une utilisation à long terme des cartes topologiques.
Donc, en plus du traitement de séquences temporelles, l'utilisation des cartes auto-organisatrices pour notre classe de problèmes doit tenir compte de l'arrivée de nouvelles données et doit préserver les anciennes connaissances déjà acquises. Pour ceci nous nous sommes intéressés aux cartes auto-organisatrices évolutives.

4.4 Les cartes auto-organisatrices évolutives

Plusieurs versions évolutives de SOM ont été proposées dans la littérature pour faire face à quelques problèmes liés à l'utilisation des cartes SOM pour le traitement de données évolutives comme le nombre de neurones qu'il faut déterminer au préalable, la nécessité de refaire l'apprentissage sur toutes les données quand de nouvelles données se présentent, etc.

FIG. *4.6: Distribution de données en dimension 2.*

Le modèle *Growing Cell Structure* (GCS) (Fritzke, 1993) possède une structure constituée d'hypertétraèdres de dimension fixée (voir la figure 4.7). Le modèle est initialisé par un hypertétraèdre. A chaque neurone est associée une erreur cumulée qui représente la somme des erreurs commises lors de la classification des entrées. Après un nombre λ d'étapes d'adaptation, l'unité q ayant le maximum d'erreurs cumulées est déterminée et une nouvelle unité est insérée. De plus, d'autres connexions sont ajoutées afin de préserver la structure d'hypertétraèdres de même dimension.

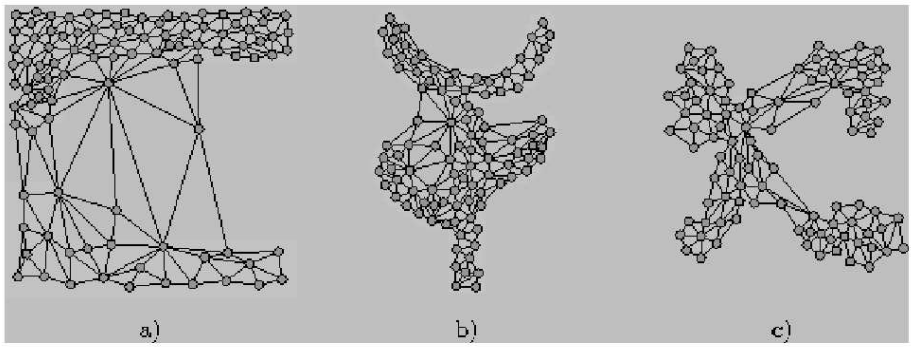


FIG. *4.7: Modèle Growing Cell Structure appliqué à plusieurs distributions de données.*

Le modèle *Growing Neural Gas (GNG)* (Fritzke, 1995b) n'impose aucune contrainte sur son graphe (voir la figure 4.8). Le graphe généré est continûment mis à jour par un apprentissage Hebbien compétitif. Celui-ci consiste, simplement, à créer une connexion entre le neurone gagnant et le second gagnant à chaque étape d'adaptation. Après un nombre fixé λ d'adaptations, l'unité q ayant l'erreur cumulée maximum est déterminée et une nouvelle unité est créée entre

q et un de ses voisins dans le graphe. Les erreurs sont ensuite redistribuées et d'autres étapes d'adaptations sont effectuées. La topologie du réseau GNG reflète celle de la distribution des signaux d'entrée.

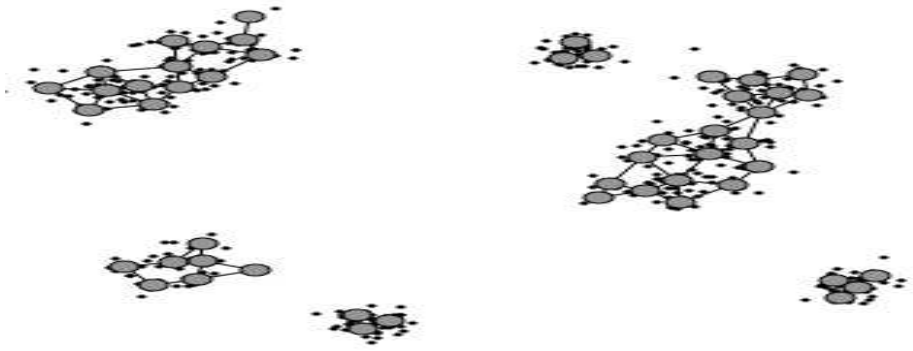


FIG. *4.8: Modèle Growing Neural Gas appliqué aux données de la figure 4.6.*

Le modèle GNG-U (Fritzke, 1995b) est une modification du modèle GNG qui permet de traiter les données non-stationnaires. Ceci est effectué en introduisant un critère qui identifie les neurones non-utiles de la carte. Dans ce modèle, les changement lents de la distribution des données sont pris en compte par l'adaptation des unités existantes et les changements rapides sont pris en compte par la suppression des neurones non-utiles et l'insertion de certains neurones dans d'autres endroits.

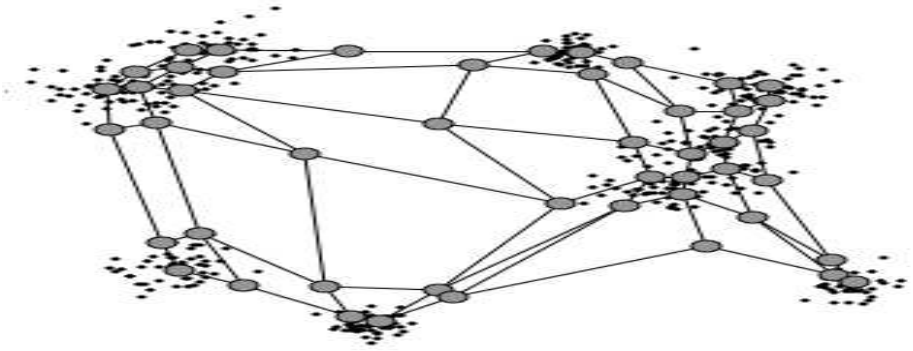


FIG. *4.9: Modèle Growing Grid appliqué aux données de la figure 4.6.*

Le modèle *Growing Grid* (GG) (Fritzke, 1995a) a une structure hyper-rectangulaire (voir la figure 4.9). La configuration initiale est un hyper-cube de dimension k. Pour maintenir cette structure, il est nécessaire d'insérer à chaque fois une ligne (ou une colonne) entière. L'adaptation

est effectuée de la même manière que dans les modèles précédents. L'erreur cumulée est aussi utilisée pour identifier le neurone au voisinage duquel une colonne ou une ligne est insérée après λ étapes d'adaptation.

Les modèles cités ont la capacité d'apprendre en continu à partir de données mais n'assurent pas la propriété de stabilité.

Le modèle *Incremental Grid Growing* (IGG) (Blackmore et Miikkulainen, 1993) est une amélioration du réseau SOM qui permet à la structure de la carte d'être dynamique en s'adaptant aux données (structure non fixée au préalable). En dimension 2, la grille est initialement constituée de quatre neurones connectés entre eux. Comme dans les versions évolutives précédentes, après un nombre λ d'étapes d'adaptation, des neurones sont ajoutés à la grille, mais cette fois au niveau du périmètre au voisinage du neurone ayant l'erreur cumulée maximale (voir la figure 4.10). Cette version ne nécessite pas un calcul préalable des dimensions de la carte, mais la même base est utilisée à chaque adaptation de la grille courante. L'aspect incrémental concerne seulement la topologie de la carte sans tenir compte de l'arrivée dynamique des données (pas de plasticité).

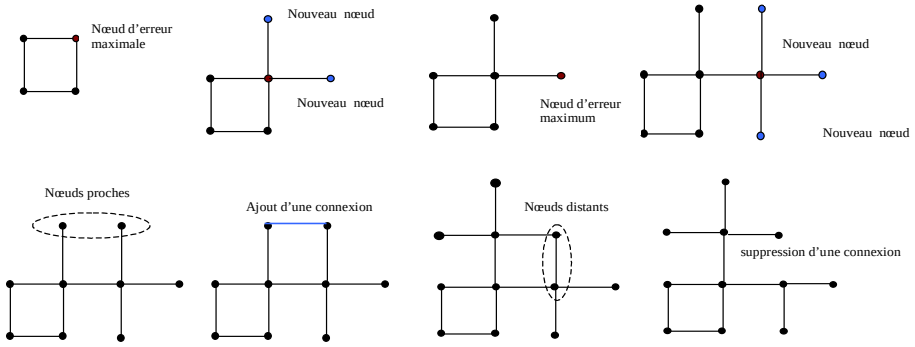


FIG. *4.10: Evolution du modèle Incremental Grid Growing.*

4.4.1 Discussion

Les modèles cités possèdent plusieurs propriétés communes. La structure de leur réseau est un graphe contenant des nœuds et des arêtes les liant. A chaque étape d'adaptation, une erreur cumulée est calculée au niveau des neurones activés. Cette erreur est utilisée pour déterminer (après un nombre d'étapes d'adaptations) l'endroit où les nouveaux neurones sont insérés. Cette variable d'erreur est ensuite redistribuée localement et de nouvelles étapes d'adaptation sont effectuées. D'autres critères d'insertion peuvent être utilisés dans les cartes auto-organisatrices. Par exemple dans le modèle incrémental *Growing Self-Organizing Map* (GSOM) (Bauer et Villmann, 1997), au lieu d'utiliser l'erreur cumulée, les auteurs proposent d'insérer les neurones

au voisinage du centre de la topologie courante de la carte. Les principales différences entre les modèles cités résident dans les contraintes imposées sur la topologie.

Certains des modèles que nous avons présentés permettent, certes, de prendre en compte l'arrivée dynamique des données mais ne garantissent pas la préservation des anciennes connaissances. Ceci peut être un inconvénient lors de leur utilisation à long terme dans certaines applications où les connaissances déjà acquises peuvent être utiles pour traiter les nouvelles données. Pour cela, nous nous sommes intéressés à la théorie de résonance adaptative (Grossberg, 1987) qui est l'un des rares modèles qui tiennent compte de l'arrivée des données tout en préservant les anciennes connaissances et à la possibilité d'utiliser ce paradigme dans les cartes auto-organisatrices.

4.5 La théorie de résonance adaptative

La théorie de résonance adaptative (ART : Adaptive Resonance Theory) a donné naissance à plusieurs modèles comme : ART1 (Grossberg, 1987), ART2 (Carpenter et Grossberg, 1988), Fuzzy ART (Carpenter *et al.*, 1991), Artmap (Carpenter, 1997), Simplified ART (Baraldi et Alpaydin, 1998), etc.
Le modèle ART1 traite uniquement les entrées binaires, ce qui est restrictif. Toutefois, d'autres modèles comme ART2, Fuzzy ART et Simplified ART remédient à ce problème en traitant les données réelles.
Dans ce paragraphe, nous allons décrire le premier modèle ART1, puis ART2 qui constitue une amélioration de ART1. Ensuite, nous allons décrire le modèle Simplified ART auquel nous nous intéressons.

4.5.1 Le système ART1

Il s'agit d'un système auto-adaptatif (apprentissage non supervisé) qui traite uniquement des entrées binaires. Il est composé de plusieurs modules dont les composants ont un comportement qui obéit à des systèmes d'équations différentielles souvent non linéaires. Ces équations comportent de nombreux paramètres à déterminer ou à ajuster. Le modèle est donc assez général dans sa formulation, toutefois les modèles opérationnels sont des sous-modèles simplifiés de ce modèle général.

4.5.1.1 Les composants du système et son fonctionnement général

Le système ART1 (Grossberg, 1987) est composé de deux modules principaux qui interagissent (voir figure 4.11) :

- *Un module d'attention MA (attentional subsystem)* qui réagit aux excitations externes et mémorise le code des évènements appris.

La partie principale de ce module est constituée d'un ensemble de mémoires à court terme (STM : Short Term Memory) et à long terme (LTM : Long Term Memory). Il possède également un mécanisme de régulation interne appelé contrôle de l'attention (CA).

– *Un module d'orientation MO (orienting subsystem)* qui permet de décider si une entrée présentée au système est nouvelle ou pas et qui, éventuellement, provoque la création d'un nouveau neurone. Le nombre de neurones est donc dynamique, ce qui constitue une particularité intéressante de ce modèle.

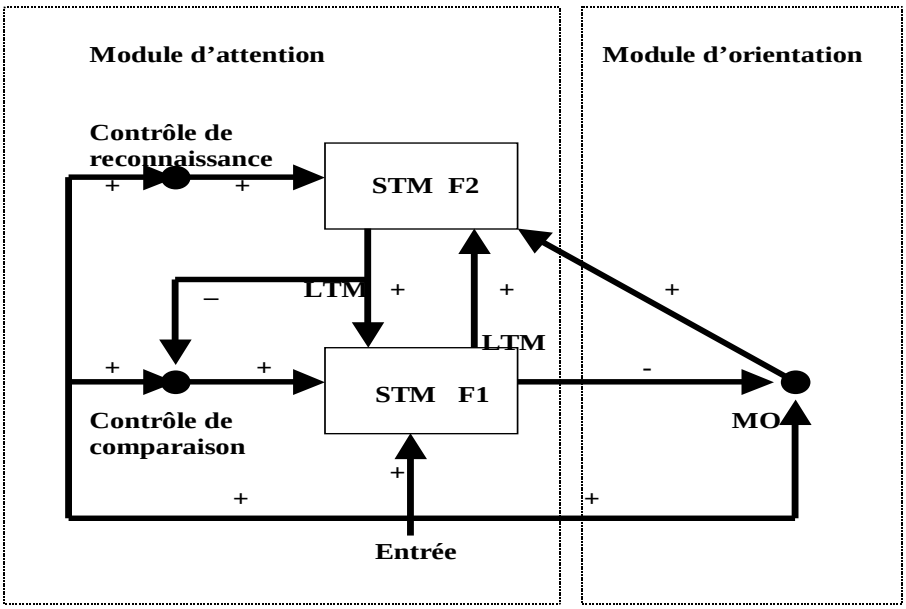


FIG. *4.11: Architecture du réseau ART1*

Module d'Attention (MA)

La partie principale de ce système est composée de :

– deux couches de neurones F_1 et F_2 (STM)
– des connexions les reliant (LTM)

De façon simplifiée, on peut dire que F_1 sert à coder l'entrée (rétine) et F_2 la catégorie de cette entrée (couche de classification). L'information circule de bas en haut ($F_1 \Rightarrow F_2$) et de haut en bas ($F_2 \Rightarrow F_1$) par un double circuit de connexions (voir la figure 4.12).

Flux $F_1 \Rightarrow F_2$
La suite de signaux produits est la suivante (voir la figure 4.12) :

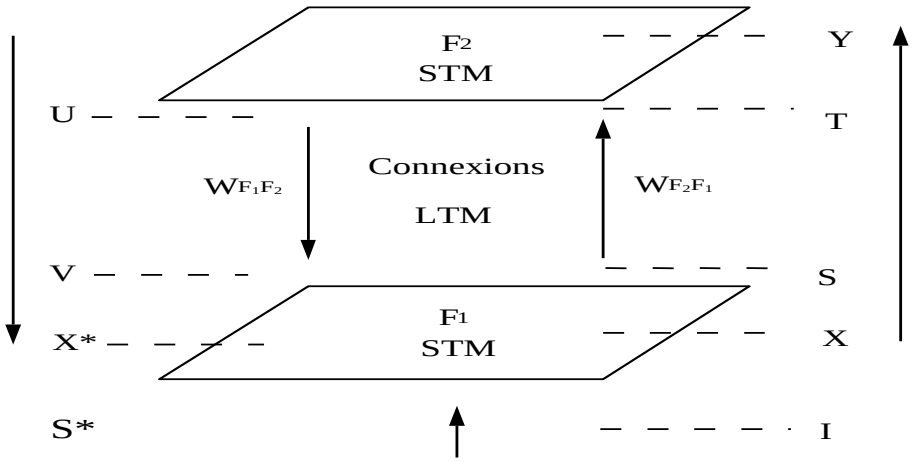


FIG. *4.12: Flux d'information entre les différentes unités de mémorisation du système ART1.*

$I \Rightarrow X \Rightarrow S \Rightarrow T \Rightarrow U$

Quand l'entrée I est présentée en F_1, elle active les neurones de F_1 qui prend un état X représentant le vecteur d'activité de ses neurones. Ces derniers émettent ensuite un signal de sortie S qui est une fonction de leur activité. S peut être considéré comme la fréquence moyenne de tir des neurones. Ce signal est envoyé sur les connexions liant F_1 à F_2 et est pondéré par les poids de ces connexions, qui modélisent un couplage avec le neurone récepteur, support de la mémoire à long terme. Cette mémoire à long terme agit comme un filtre adaptatif et produit à partir de S un signal T envoyé en entrée de F_2 : une composante de T sera comme d'habitude la somme pondérée des composantes de S par les poids des connexions arrivant en cette cellule de F_2. L'évolution dynamique de F_2 produit alors un état stationnaire : le signal U qui est rangé dans la STM de F_2.

Flux $F_2 \Rightarrow F_1$

La suite de signaux produits est la suivante :

$U \Rightarrow V \Rightarrow X* \Rightarrow S*$

F_2 envoie le signal U vers F_1 à travers le deuxième système de la LTM : les connexions $F_2 \Rightarrow F_1$.

Par passage dans ce deuxième filtre adaptatif, U est transformé en V qui est une somme pondérée de U par les poids des connexions $F_2 \Rightarrow F_1$.

V et I sont alors comparés dans F_1 pour produire l'activité $X*$ sur les neurones de F_1.

$X*$ est une mesure de la différence entre l'entrée I et le représentant en mémoire V correspondant à la catégorie où le système a classé I durant cette phase d'aller-retour dans la mémoire.

$X*$ est transformé en $S*$ qui code la forme que le système a reconnue dans l'espace d'entrée.

Si $S*$ et I sont proches, F_1 entre en "résonance", c'est cette propriété qui donne son nom au système.

Quand une forme I est présentée en F_1, si le neurone activé en F_2 correspond à une forme mémorisée éloignée de I alors $X*$ va être grand (en norme).

Dans ce cas, on inhibe pour la suite le noeud de F_2 qui vient d'être sélectionné et on recommence

un aller-retour dans le système, ce qui va conduire à sélectionner une autre forme dans F_2 et ainsi de suite.
Ce cycle pourra se terminer si :

- le réseau a trouvé dans sa mémoire une forme proche de I
- la mémoire a été explorée sans succès, le mécanisme précédent sélectionne alors un neurone vierge de F_2 pour coder une nouvelle catégorie dont l'entrée I sera le premier représentant.

Dans ces deux cas, on modifie les connexions de la LTM pour réaliser l'apprentissage de l'entrée I.

D'autres signaux circulent dans ce système dans le but de le rendre entièrement autonome, nous avons mentionné le contrôle de l'attention, qui envoie des signaux sur F_1 et F_2.
Ils permettent à la couche F_1 de distinguer la provenance des signaux qu'elle reçoit (de l'extérieur : les signaux I, ou de F_2 : les signaux V). Dans F_2, ce signal d'attention provoque la décroissance de l'activité des cellules, ce qui permet à F_2 de ré-enclancher sa propre dynamique quand une nouvelle forme T lui est présentée.

Module d'Orientation (MO)
Les cycles de recherche d'une forme en mémoire qui viennent d'être décrits sont automatisés par les signaux envoyés par une autre composante de ART : le module d'orientation (MO). Il permet d'inhiber pendant une recherche en mémoire les neurones de F_2 et d'éliminer ainsi pour la suite de la phase de recherche des catégories jugées non représentatives. L'existence de ce système est un exemple du souci de Grossberg de construire des systèmes complets et autonomes.
Deux autres facteurs interviennent pour le codage des formes en différentes catégories :

Le seuil de vigilance ρ :
Ce paramètre détermine une distance seuil au delà de laquelle une entrée I et une forme reconnue V seront dites différentes. Ce facteur de vigilance permet de régler la finesse des catégories. En l'augmentant, on induit une classification plus fine des entrées. Ce paramètre doit être fourni au système.

Le contexte :
La prise en compte du contexte est faite automatiquement. Pour un niveau de vigilance donné, deux formes simples qui se différencient par k bits seront classées dans deux catégories différentes, alors que deux formes plus complexes distantes également de k bits pourront être classées dans la même catégorie.

Le système est donc capable de voir si la distance entre les formes est significative ou pas.
Le système fonctionne de façon à ce que l'activation de la couche F_1 soit soumise à *la Règle des*

2/3 : deux des trois sources de signal (I, V, contrôle de l'attention) doivent être présents pour que F_1 soit activé.

4.5.1.2 Dynamique des différents modules

Les deux couches F_1, F_2 et la LTM possèdent leur propre dynamique décrite par des équations différentielles.

Module d'Attention

F_1 et F_2 évoluent suivant une équation dite de la membrane qui a la forme générale suivante :

$$\epsilon \frac{dx_k}{dt} = -x_k + (1 - Ax_k)J_k^+ - (B + Cx_k)J_k^- \tag{4.17}$$

où x_k représente l'état d'une cellule k en F_1 ou F_2, J_k^+ est la somme des entrées excitatrices de cette cellule et J_k^- la somme de ses entrées inhibitrices. Les paramètres A, B, C sont positifs et ϵ contrôle la rapidité d'évolution.

Les notations suivantes sont utilisées dans la suite :
x_{1i} : activation de l'unité v_i de la couche F_1
x_{2j} : activation de l'unité u_j de la couche F_2
w_{ij} : connexion de l'unité u_j de F_2 vers l'unité v_i de F_1

Dynamique sur F1

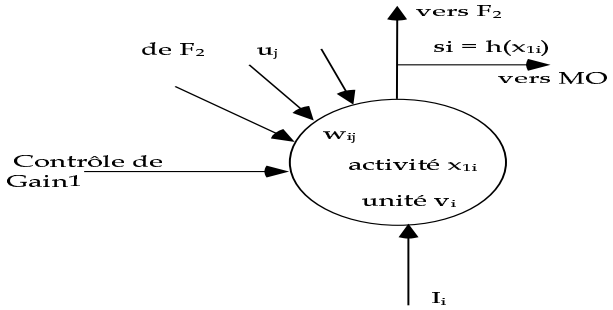


FIG. *4.13: Dynamique su* F_1.

Chaque neurone calcule une entrée en provenance de F_2 :

$$v_i = \sum_j u_j w_{ij} \tag{4.18}$$

calcule sa sortie s_i :

$$s_i = h(x_{1i}) = \begin{cases} 1 & si\ x_{1i} > 0 \\ 0 & si\ x_{1i} \leq 0 \end{cases} \tag{4.19}$$

la somme des entrées excitatrices de ce neurone (J_i^+) est donnée par :

$$J_i+ = I_i + D_1 V_i + B_1 G \tag{4.20}$$

le terme inhibiteur J_k^- dirige le signal de contrôle de l'attention qui est égal à 1, D_1 et B_1 sont des constantes.

La dynamique de F_1 est donc :

$$\epsilon \frac{dx_{1i}}{dt} = -x_{1i} + (1 - A_1 x_{1i})(I_i + D_1 V_i + B_1 G) - (B_1 + C_1 x_{1i}) \tag{4.21}$$

avec :
D_1, B_1 et C_1 sont des constantes et

$$G = \begin{cases} 1 & si\ I \neq 0\ et\ U = 0 \\ 0 & sinon \end{cases} \tag{4.22}$$

Dynamique sur F2

Chaque unité calcule une entrée en provenance de F_1 :

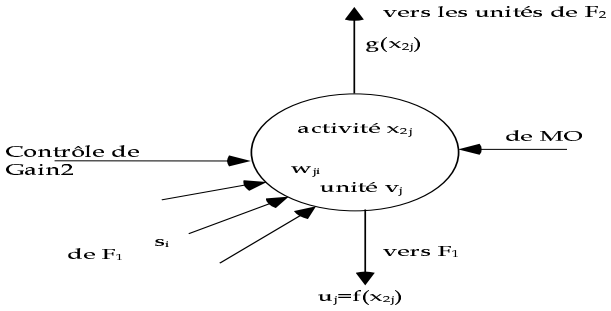


FIG. *4.14: Dynamique sur F_2.*

$$T_j = \sum_i s_i w_{ji} \tag{4.23}$$

les sommes des entrées excitatrices (J_j^+) et inhibitrices (J_j^-) de cette cellule sont données par :

$$J_j^+ = D_2 T_j + g(x_{2j}) \tag{4.24}$$

$$J_j^- = \sum_{k \neq j} g(x_{2k}) \tag{4.25}$$

où D_2 est une constante. La dynamique de F_2 est donc :

$$\epsilon \frac{dx_{2j}}{dt} = -x_{2j} + (1 - A_2 x_{2j})(D_2 T_j + g(x_{2j})) - (B_2 + C_2 x_{2j}) \sum_{k \neq j} g(x_{2k}) \tag{4.26}$$

où A_2, B_2, C_2 et D_2 sont des constantes. Les sorties de F_2 sont calculées par :

$$u_j = f(x_{2j}) = \begin{cases} 1 & si\ T_j = \max_k \{T_k\} \\ 0 & sinon \end{cases} \tag{4.27}$$

Dynamique des poids $F_2 \Rightarrow F_1$: Top-Down LTM

Les poids synaptiques $F_2 \Rightarrow F_1$ obéissent à une équation de la forme :

$$\frac{dw_{ij}}{dt} = [-w_{ij} + h(x_{1i})](x_{2j}) \tag{4.28}$$

$$\frac{dw_{ij}}{dt} = \begin{cases} -w_{ij} + 1 & si\ u_j\ active\ et\ v_i\ active \\ -w_{ij} & si\ u_j\ active\ et\ v_i\ inactive \\ 0 & si\ u_j\ inactive\ et\ v_i\ inactive \end{cases} \tag{4.29}$$

Les valeurs asymptotiques des connexions (si v_J est l'unité gagnante) :

$$w_{iJ} = \begin{cases} 1 & si\ v_i\ active \\ 0 & sinon \end{cases} \tag{4.30}$$

Dynamique des poids $F_1 \Rightarrow F_2$: Bottom-Up LTM

Les poids synaptiques $F_1 \Rightarrow F_2$ obéissent à une équation de la forme :

$$\frac{dw_{ji}}{dt} = K f(x_{2j})[(1 - w_{ji}) L h(x_{1i}) - w_{ji} \sum_{k \neq i} h(x_{1k}] \tag{4.31}$$

où K et L sont des constantes, (x_{2j}) est la sortie de u_j, et $h(x_{1i})$ est la sortie de v_i.
Les notation suivantes sont utilisée dans ce qui suit :
I : l'entrée (vecteur binaire)
et $|I| = \sum_i I_i$
S : sortie de F_1
et $|S| = \sum_i h(x_{1i})$

$$\sum_{k \neq i} h(x_{1k}) = \sum_k h(x_{1k}) - h(x_{1i}) = \begin{cases} |S| - 1 & si\ v_i\ active \\ |S| & si\ v_i\ inactive \end{cases} \tag{4.32}$$

Alors

$$\frac{dw_{ji}}{dt} = \begin{cases} K[(1 - w_{ji})L - w_{ji}(|S| - 1)] & si\ u_i\ active\ et\ v_j\ active \\ -K[w_{ji}|S|)] & si\ u_i\ inactive\ et\ v_j\ active \\ 0 & si\ v_j\ inactive \end{cases} \tag{4.33}$$

où L et K sont des constantes. Les valeurs asymptotiques des connexions (si v_J est l'unité gagnante) :

$$w_{Ji} = \begin{cases} \frac{L}{L-1+|S|} & si\ v_i\ active \\ 0 & sinon \end{cases} \tag{4.34}$$

Le réseau ART1 effectue une auto-organisation et une auto-stabilisation comme réponse à des entrées binaires. L'algorithme détaillé du modèle ART1 est décrit dans l'annexe A.
Le principe de l'algorithme du modèle ART1 est décrit de façon simplifié ci-dessous.

Quand une nouvelle entrée ξ est présentée à la couche de reconnaissance, la valeur d'activation des neurones de la couche de comparaison est calculée en utilisant une fonction d'activation unidirectionnelle $\vec{F}(,)$[7]. C'est une mesure de similarité non symétrique. Un apprentissage compétitif est ensuite effectué. Le neurone gagnant (éligible à être résonant) $s(\xi)$ est la solution, si elle existe, du problème de maximisation suivant :

$$s(\xi) = \arg\max_{c \in A} \vec{F}(\xi, w_c) \tag{4.35}$$

soumis au test de vigilance décrit par :

$$\vec{G}(w_s, \xi) \geq \rho, \quad \rho \in [0,1] \tag{4.36}$$

Où A est l'ensemble des neurones et w_c le poids du neurone c, ρ est le seuil de vigilance défini par l'utilisateur et $\vec{G}(,)$[8] est une fonction de similarité unidirectionnelle (non symétrique).
Si le test de vigilance n'est pas satisfait (la résonance ne se produit pas), le neurone gagnant est inhibé et une recherche du second gagnant est effectuée. Celui-ci est soumis à son tour au test de vigilance.
Si le test de vigilance est satisfait, l'activité du neurone gagnant est renforcée en ajustant son vecteur de poids pour le rapprocher de l'entrée. Si aucune solution n'existe pour le problème de maximisation décrit précédemment, un nouveau neurone est dynamiquement alloué pour la nouvelle entrée.

REMARQUE. Dans ART1 (Carpenter et Grossberg, 1987) et (Carpenter et Grossberg, 1988), le test de vigilance utilise la fonction $\vec{G}(,)$ tandis que le neurone éligible pour être résonant est recherché suivant la valeur de la fonction $\vec{F}(,)$. Donc, le neurone gagnant en utilisant $\vec{F}(,)$ n'est pas nécessairement le gagnant en utilisant $\vec{G}(,)$, ceci justifie la recherche qui est répétée jusqu'à ce que le test de vigilance soit satisfait.

[7] $\vec{F}(,)$ correspond à la fonction T_j ($T_j = \sum_{i=1}^{m} s_i w_{ji}$) dans l'algorithme ART1 de l'annexe A.
[8] $\vec{G}(,)$ correspond à la fonction définie par : $\frac{|S|}{|I|} = \frac{\sum_{i=1}^{m} S_i}{\sum_{i=1}^{m} I_i}$ dans l'algorithme ART1 de l'annexe A.

4.5.2 Le système ART2

4.5.2.1 Principe

ART2 (Carpenter et Grossberg, 1988) possède les mêmes composants que ART1 dont il constitue une extension au traitement des entrées à composantes réelles.
Ceci est rendu possible par une complexification de la couche de comparaison F_1 et l'utilisation de dynamiques différentes.
La couche F_1 possède maintenant trois niveaux de calcul et de systèmes de contrôle de l'attention (voir la figure 4.15).
Les signaux d'entrée et ceux venant de F_2 arrivent à différents endroits de F_1 (respectivement les niveaux bas et haut), la comparaison entre I et V se fait également sur un autre niveau de F_1 (niveau intermédiaire).
La couche F_1 est divisée en six niveaux : $z,\ x,\ u,\ v,\ p,\ q$. Sur chacun de ces niveaux, les activités des cellules sont normalisées de façon à pouvoir être comparées.
F_1 réalise plusieurs mises en correspondance. Des boucles "feedback" lui permettent d'augmenter le contraste et de supprimer le bruit sur les formes traitées.

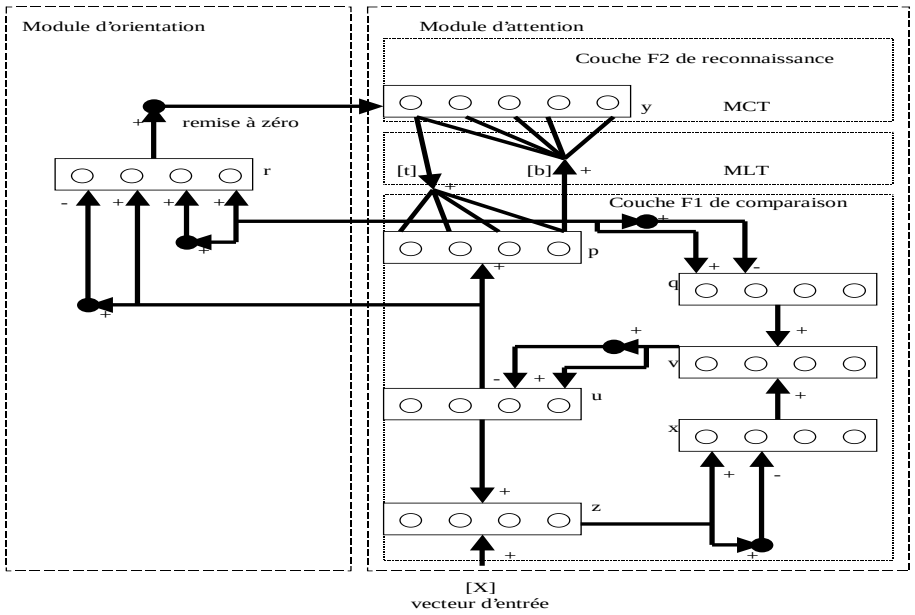


FIG. *4.15: Architecture du réseau ART2.*

4.5.2.2 Dynamique des différents modules

La dynamique des couches F_1 et F_2 obéit à des équations qui ont la même forme générale que celles de ART1.

Évolution sur F_1 :

Les activations des unités de F_1 évoluent suivant l'équation suivante :

$$\epsilon \frac{dx_k}{dt} = -Ax_k + (1 - Bx_k)J_k^+ - (C + Dx_k)J_k^- \tag{4.37}$$

où A, B, C et D sont des constantes.

Pour simplifier le modèle ART2, nous prenons B=C=0 et nous considérons la solution asymptotique :

$$x_k = \frac{J_k^+}{A + DJ_k^-} \tag{4.38}$$

Les valeurs des différents termes de cette équation varient selon les différentes sous-couches (voir le tableau 4.2) :

Couche	A	D	J_i^+	J_i^-	Activation
z	1	1	$I_i + au_i$	0	$z_i = I_i + au_i$
x	e	1	z_i	$\|\|z\|\|$	$x_i = \frac{z_i}{e+\|\|z\|\|}$
u	e	1	v_i	$\|\|v\|\|$	$u_i = \frac{v_i}{e+\|\|v\|\|}$
v	1	1	$f(x_i) + bf(q_i)$	0	$v_i = f(x_i)+bf(q_i)$
p	1	1	$u_i + \sum_j g(y_i)w_{ij}$	0	$p_i = u_i + \sum_j q(y_i)w_{ij}$
q	e	1	p_i	$\|\|p\|\|$	$q_i = \frac{p_i}{e+\|\|p\|\|}$
r	e	1	$u_i + cp_i$	$\|\|u\|\| + c\|\|p\|\|$	$r_i = \frac{u_i+cp_i}{e+\|\|u\|\|+c\|\|p\|\|}$

TAB. *4.2: Les valeurs des différents termes de l'équation 4.37 selon les différentes couches de* F_1.

Évolution sur F_2 :

Chaque unité calcule une entrée en provenance de F_1 :

$$T_j = \sum_i p_i w_{ji} \tag{4.39}$$

Les sorties de F_2 sont calculées par :

$$g(y_j) = \begin{cases} d & si\ T_j = \max_k\{T_k\} \\ 0 & sinon \end{cases} \tag{4.40}$$

On peut réécrire les équations de la sous-couche p :

$$p_i = \begin{cases} u_i & si\ F_2\ inactive \\ u_j + dw_{iJ} & si\ la\ cellule\ J\ de\ F_2\ est\ active \end{cases} \tag{4.41}$$

Évolution des poids $F_1 \Rightarrow F_2$ (Bottom-Up) :

$$\frac{dw_{ji}}{dt} = g(y_j)(p_i - w_{ji}) \tag{4.42}$$

Évolution des poids $F_2 \Rightarrow F_1$ (Top-Down) :

$$\frac{dw_{ij}}{dt} = g(y_j)(p_i - w_{ij}) \tag{4.43}$$

Pour plus de détails sur l'algorithme ART2, voir l'annexe A.

4.5.3 Le modèle *Simplified ART*

Le modèle *Simplified ART* (SART) (Baraldi et Alpaydin, 1998) a été conçu pour faire face à certaines limites du modèle ART1 et permet de simplifier l'architecture des modèles ART. Ce modèle est capable de traiter des entrées multidimensionnelles à valeurs réelles. Étant donné un vecteur d'entrée à valeurs réelles $\xi \in R^n$ (aucun pré-traitement n'est effectué), le neurone gagnant $s(\xi)$ est la solution, si elle existe, du problème de maximisation suivant :

$$s(\xi) = \arg\max_{c \in A} \overline{F}(\xi, w_c) \tag{4.44}$$

soumis à la contrainte de vigilance :

$$\overline{G}(w_s, \xi) \geq \rho, \quad \rho \in [0, 1] \tag{4.45}$$

où $\overline{F}(,)$ and $\overline{G}(,)$ sont des fonctions d'activation et de similarité (respectivement) symétriques et sont choisies de façon que :

$$\overline{G}(w_{c_1}, \xi) > \overline{G}(w_{c_2}, \xi) \Rightarrow \overline{F}(\xi, w_{c_1}) > \overline{F}(\xi, w_{c_2}) \tag{4.46}$$

et vice versa.

Le processus de recherche n'est pas nécessaire pour détecter le domaine de résonance : quand le vecteur de poids du neurone gagnant w_s, sélectionné selon $\overline{F}(,)$, ne satisfait pas la contrainte de vigilance, un nouveau neurone peut être immédiatement alloué pour ξ. De plus, Comme les fonctions $\overline{F}(,)$ et $\overline{G}(,)$ sont symétriques, en prenant $\overline{F}(,) = \overline{G}(,)$ la propriété 4.46 est vérifiée.

4.5.4 Discussion

Les modèles issus de la théorie de résonance adaptative présentés ont été conçus pour doter les réseaux de neurones des propriétés de plasticité et de stabilité. L'utilisation du réseau ART1 est restreinte puisqu'il ne traite que des entrées binaires, ceci représente un inconvénient majeur de ce modèle.
Pour faire face à ce problème, d'autres modèles comme ART2 et Simplified ART qui traitent des entrées réelles ont été proposés.
La structure du modèle ART2 est complexe et difficile à mettre en œuvre.
Le modèle Simplified ART (SART) est une variante simplifiée des modèles précédents tout en gardant les principes de la théorie de résonance adaptative, ceci représente un point fort de ce modèle. Pour ceci, nous avons choisi l'intégration du modèle SART dans une carte topologique.

Nous avons étudié les cartes auto-organisatrices temporelles, les cartes incrémentales et les réseaux ART dans le but de concevoir un réseau de neurones qui traite les séquences temporelles et qui possède les propriétés de stabilité et de plasticité du réseau ART nécessaires pour faire l'hybridation avec le système « CASEP ».
Dans la suite, nous proposons de nouveaux modèles de cartes auto-organisatrices qui possèdent ces propriétés.

Chapitre 5

Nouveaux modèles de cartes auto-organisatrices

Dans le but de construire une carte auto-organisatrice qui traite les séquences temporelles et qui possède les propriétés de plasticité et de stabilité, nous proposons plusieurs nouveaux modèles de cartes auto-organisatrices. Nous décrivons dans ce chapitre l'aspect conceptuel et algorithmique de ces modèles ainsi que les expérimentations effectuées pour les valider.

5.1 Incorporation des propriétés de plasticité et de stabilité dans SOM

Dans ce modèle, nous nous sommes intéressés à l'utilisation des cartes auto-organisatrices pour le traitement des données évolutives. Les réseaux de neurones artificiels « statiques » comme la carte SOM (*Self Organizing Map*) ne permettent pas d'acquisition de nouvelles connaissances (plasticité) tout en maintenant les anciennes (stabilité). Plusieurs versions des cartes auto-organisatrices évolutives comme les modèles *Growing Cell Structure (GCS)* et *Growing Neural Gas (GNG)* ont été proposées pour acquérir en continu de nouvelles connaissances, mais ne tiennent pas compte des anciennes déjà acquises.
Le compromis entre la plasticité et la stabilité est souvent appelé : « le dilemme stabilité-plasticité ». La théorie de la résonance adaptative (ART : *Adaptive Resonance Theory*) est spécialement conçue pour faire face à ce dilemme. Les modèles ART sont parmi les rares réseaux de neurones artificiels possédant les deux propriétés. Nous nous sommes intéressés particulièrement à la classe « Simplified ART ».
Nous présentons dans ce paragraphe une nouvelle version des cartes auto-organisatrices baptisée : « SOM-ART » qui effectue des tâches de classement et de classification (Zehraoui et Bennani, 2004c). Cette nouvelle version possède les propriétés de stabilité et de plasticité. Le modèle SOM-ART intègre une carte SOM évolutive dans un paradigme basé sur la théorie de la résonance adaptative. Le modèle *Simplified ART* (SART) (Baraldi et Alpaydin, 1998) a été

conçu pour faire face à certaines limites du modèle ART1 et permet de simplifier l'architecture des modèles ART. Ce modèle est capable de traiter des entrées multidimensionnelles à valeurs réelles.

Nous avons intégré une carte auto-organisatrice incrémentale dans un modèle *Simplified ART*. SOM-ART est une carte topologique de voisinage carré (même voisinage que dans IGG, voir la figure 5.1) , mais les critères d'initialisation de la carte et d'insertion de neurones sont différents. Ceux-ci sont effectués suivant le paradigme ART. En plus de la notion d'auto-organisation qui existe dans les réseaux *Simplified ART*, SOM-ART permet de projeter des données de dimension quelconque dans un espace discret de façon que les données similaires aient des projections proches dans la carte. Ceci est l'une des particularité des cartes SOM qui facilite la visualisation des données et leur interprétation.

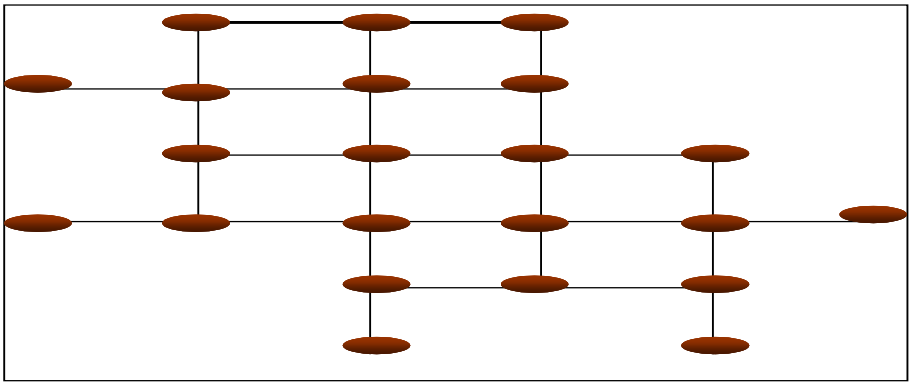


FIG. *5.1: La carte SOM-ART.*

Le tableau 5.1 montre les propriétés de notre modèle SOM-ART en comparaison avec différentes cartes SOM ainsi qu'avec le réseau Simplified ART. Le symbole '+' représente la présence de la propriété dans le réseau ou dans certaines variantes du réseau et '-' son absence.

Nous remarquons (voir le tableau 5.1) que notre modèle SOM-ART possède toutes les propriétés (stabilité, plasticité, classement, classification et visualisation) contrairement aux autres modèles auxquels nous l'avons comparé. Les modèles SART n'effectuent pas de classement et surtout ne permettent pas la visualisation des données. Les cartes SOM ne possèdent pas les propriétés de stabilité et de plasticité. Les versions évolutives de SOM ne possèdent pas la propriété de stabilité.

	SOM-ART	SART	SOM	SOM évolutives
Stabilité	+	+	-	-
Plasticité	+	+	-	+
Classement	+	-	+	+
Classification	+	+	+	+
Visualisation	+	-	+	+

TAB. *5.1: Comparaison des propriétés des différentes cartes SOM et du réseau Simplified ART.*

Algorithme d'apprentissage de SOM-ART

La carte SOM-ART est d'abord initialisée à un neurone, ensuite des neurones sont ajoutés suivant les critères du réseau *Simplified ART* en utilisant le test de vigilance pour sélectionner le neurone gagnant. Si le test de vigilance est satisfait, une mise à jour de la carte est effectuée. Sinon, un neurone est ajouté au périmètre de la carte comme dans IGG mais cette fois au voisinage du neurone le plus proche de l'entrée qui se trouve dans le périmètre de la carte tout en respectant la structure de la carte (voisinage carré). Des connexions sont aussi ajoutées à la carte pour lier le neurone ajouté à ses voisins.
Durant la dernière étape d'apprentissage, les neurones de la carte sont étiquetés en utilisant un vote majoritaire sur l'ensemble des étiquettes des entrées qui les ont activés.
La distance euclidienne $dist(,)$ (Baraldi et Alpaydin, 1998) est utilisée pour comparer les neurones. Les fonctions d'activation et de similarité, décrites précédemment (voir le paragraphe §4.4), du réseau SOM-ART sont obtenues à partir de la distance euclidienne $dist(,)$ comme suit :

$$\overline{F}(,) = \overline{G}(,) = \frac{1}{1 + dist(,)} \tag{5.1}$$

Ces fonctions sont symétriques, donc vérifient la propriété (4.46) et peuvent être utilisées dans SOM-ART.
Les fonctions $\overline{F}(,)$ et $\overline{G}(,)$ sont définies de $R^n \times R^n$ dans $]0,1]$. Elles sont inversement proportionnelles à la distance euclidienne. Donc, pour toute entrée ξ, nous avons :

$$\arg\max_{c \in A} \overline{F}(\xi, w_c) = \arg\min_{c \in A} dist(\xi, w_c) \tag{5.2}$$

L'algorithme 5 décrit le processus d'apprentissage de SOM-ART.

Algorithme 5 Algorithme d'apprentissage de SOM-ART.

Initialiser l'ensemble A des neurones à une seule unité c_1.

Initialiser le paramètre temps t : $t = 0$.

Initialiser l'ensemble des connexions C, $C \subset A \times A$ à l'ensemble vide : $C = \emptyset$.

Tant que $(t \leq t_{max})$ **faire**

1. Pour toute entrée ξ de la base d'apprentissage
   Le vecteur référence w_{c1} de la première unité c_1 est initialisé a la première entrée.
   - Déterminer le gagnant $s(\xi) \in A$ par :

$$s(\xi) = \arg\min_{c \in A} dist(\xi, w_c)$$

   - Soumettre le neurone gagnant $s(\xi)$ au test de vigilance :
     - Si $(\frac{1}{1+dist(s(\xi),\xi)} \geq \rho)$, adapter chaque unité r selon :

$$\Delta w_r = \epsilon(t) h_{rs} [\xi - w_c]$$

     où $\epsilon(t) = \epsilon_i (\frac{\epsilon_f}{\epsilon_i})^{\frac{t}{t_{max}}}$, $h_{rs} = \exp(\frac{-d_1(r,s)^2}{2\sigma^2})$ et $\sigma(t) = \sigma_i (\frac{\sigma_f}{\sigma_i})^{\frac{t}{t_{max}}}$
     $\epsilon(t)$ est le pas d'apprentissage, ϵ_i et ϵ_f sont les valeurs initiale et finale de ϵ, σ_i et σ_f sont les valeurs initiale et finale de σ, $d_1(,)$ est la distance de Manhattan et h_{rs} est la fonction de voisinage gaussienne.
     - Sinon ajouter une nouvelle unité c_r dans le voisinage du neurone le plus proche de l'entrée dans le périmètre de la carte, son vecteur référence w_r est initialisé par : $w_r = \xi$.
2. Si $(t = t_{max})$ étiqueter chaque unité en utilisant un vote majoritaire sur les étiquettes des entrées qui l'ont activée.
3. Incrémenter le paramètre temps t : $t = t + 1$

La validation de ce modèle est décrite dans le paragraphe 5.4.3. Nous présentons dans la suite le modèle M-SOM qui prend en compte l'aspect temporel des données.

5.2 Prise en compte de l'aspect temporel dans les cartes auto-organisatrices

Nous présentons dans ce paragraphe, des approches pour la classification et le classement de séquences (Zehraoui et Bennani, 2004b). Les approches que nous proposons utilisent des cartes auto-organisatrices (SOM : *Self Organizing Map*). Différents modèles sont utilisés pour représenter les entrées de la carte. Ces modèles permettent de tenir compte des informations contenues

dans les séquences. Les premières approches que nous proposons représentent les entrées de la carte par un vecteur représentatif ou par une matrice de covariance afin de prendre en compte les corrélations existantes entre les composants des séquences. Ces approches ne prennent pas en compte l'ordre temporel contenu dans ces séquences. Les autres approches proposées introduisent une dynamique temporelle dans la matrice de covariance. Lorsque les séquences sont représentées par des matrices de covariance, SOM est modifiée afin de prendre en compte le fait que les entrées sont des matrices et pas des vecteurs. Les expérimentations montrent que les approches que nous proposons sont meilleures que d'autres cartes auto-organisatrices temporelles pour le classement des utilisateurs d'un site Web de *e-commerce*.
Une séquence X, dans notre travail, est un ensemble fini de P_X vecteurs ordonnés $x_i \in R^n$ ($1 \leq i \leq P_X$, où P_X est la longueur de la séquence X). Dans les approches que nous proposons, l'information temporelle est représentée de manière externe : avant l'utilisation de la carte SOM, la séquence d'entrée est modélisée en utilisant sa matrice de covariance.
Dans ces approches, les cartes auto-organisatrices proposées effectuent des tâches de classification comme dans SOM. Le classement des séquences est réalisé en étiquetant les neurones de la carte : durant la dernière étape de la phase d'apprentissage, chaque neurone est étiqueté en utilisant un vote majoritaire sur les étiquettes des entrées qui l'ont activé. Si un neurone n'est pas activé, le vote majoritaire est effectué sur les étiquettes de ses voisins les plus proches.

5.2.1 Approche 1 : Modélisation par vecteurs propres (VectSOM)

Dans cette approche (voir la figure 5.2), chaque séquence est modélisée par une concaténation des deux vecteurs propres associés aux plus grandes valeurs propres de sa matrice de covariance et du vecteur moyenne de tous les vecteurs de la séquence. La moyenne donne la position du nuage de points qui représente la séquence et les deux vecteurs propres sa forme. Ce modèle est utilisé pour l'identification des locuteurs (Bennani *et al.*, 1990). Il permet de résumer l'information contenue dans toute la séquence. Le problème des longueurs variables des séquences ne se pose pas puisque toutes les séquences sont représentées par des vecteurs de même dimension.

Pour chaque séquence $X = x_1, x_2, ..., x_{P_X}$, la matrice de covariance COV_X qui lui est associée est donnée par :

$$COV_X = \frac{1}{P_X} \sum_{i=1}^{P_X} (x_i - \bar{x})(x_i - \bar{x})^T \tag{5.3}$$

où $\bar{x} = \frac{1}{P_X} \sum_{i=1}^{P_X} x_i$ est le vecteur moyenne de la séquence X et x^T est la transposée du vecteur x.

L'entrée ξ du réseau SOM est la concaténation des vecteurs propres e_1 et e_2 associés aux plus

grandes valeurs propres[9] de la matrice de covariance COV_X et le vecteur moyenne $\bar{x}$ de tous les vecteurs de la séquence : $\xi = [\bar{x}; e_1; e_2]$.

Pour comparer les modèles, nous avons utilisé la distance euclidienne. L'algorithme d'apprentissage est celui de SOM décrit précédemment (voir le paragraphe 2).

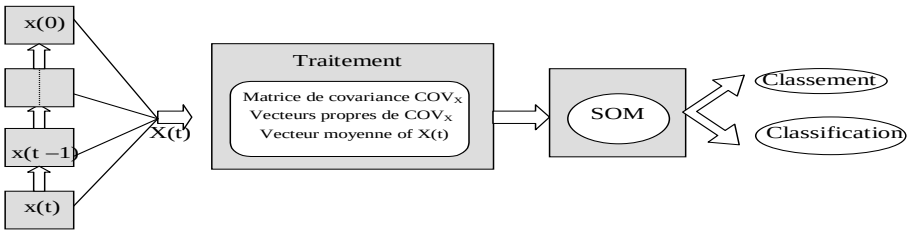


FIG. *5.2: Approche utilisant les vecteurs propres et le vecteur moyenne.*

5.2.2 Approche 2 : Modélisation par une matrice de covariance (M-SOM (COV))

Cette approche est une généralisation de la première (voir la figure 5.3). Au lieu de représenter la séquence par une concaténation de vecteurs, toute séquence est modélisée par sa matrice de covariance. Cette matrice permet de représenter la position (puisque la moyenne intervient dans le calcul de la matrice de covariance) et la forme du nuage de points qui représente la séquence. Cette information est plus riche que celle représentée dans la première approche. Comme dans l'approche précédente, tous les modèles des séquences sont de même dimension.
Afin de prendre en compte la modélisation des entrées sous forme de matrices, nous avons généralisé la carte de Kohonen, qui prend des vecteurs comme entrées.

Pour chaque séquence $X = x_1, x_2, ..., x_{P_X}$, l'entrée de la carte est $COV_X \in R^n \times R^n$. Les poids des neurones $W_c \in R^n \times R^n,\ c \in A$ (A est l'ensemble de neurones) sont représentés par des matrices ayant la même dimension que celle de l'entrée.

La distance entre une matrice de covariance $COV_X = (x_{ij}),\ 1 \leq i, j \leq n$ et les poids du neurone $W_c = (w^c_{ij}),\ 1 \leq i, j \leq n$ est la distance matricielle de Frobenius (fd) donnée par :

$$fd(COV_X, W_c) = [tr(COV_X - W_c)^T(COV_X - W_c)]^{1/2} = [\sum_i \sum_j (x_{ij} - w^c_{ij})^2]^{1/2} \qquad (5.4)$$

Cette distance est proche de la distance euclidienne entre les vecteurs. Comme dans SOM, le neurone gagnant $s(X)$ de la carte est sélectionné en utilisant cette distance comme suit :

[9]Le choix du nombre valeurs propres (et donc celui des vecteurs propres) est effectué en calculant l'inertie.

$$s(X) = \arg\min_{c \in A} fd(COV_X, W_c) \tag{5.5}$$

où A est l'ensemble des neurones.
La règle d'adaptation est donnée par l'équation :

$$W_c(t+1) = W_c(t) - \epsilon(t) h_{c,s(X)}(COV_X - W_c(t)) \tag{5.6}$$

où $h_{c,s(X)}$ est la fonction de voisinage et $\epsilon(t)$ est le pas d'apprentissage.

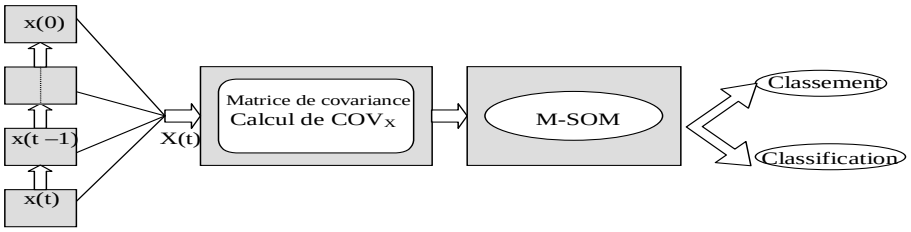


FIG. *5.3: Approche utilisant la matrice de covariance.*

Les approches que nous avons proposées représentent les vecteurs qui composent la séquence, mais elles ne prennent pas en compte la dynamique temporelle contenue dans ces séquences (nous obtenons la même matrice de covariance et le même vecteur moyenne pour tout changement de l'ordre des vecteurs de la séquence). Dans ce qui suit, nous proposons quelques modifications dans la modélisation des séquences pour prendre en compte l'ordre des vecteurs dans la séquence.

5.2.3 Approche 3 : Modélisation par une matrice de covariance pondérée (M-SOM (WCOV))

Cette approche consiste à représenter la séquence d'entrée par une matrice de covariance pondérée. Nous avons introduit un poids λ dans la matrice de covariance afin de représenter la dynamique temporelle dans la séquence et de déterminer l'influence des vecteurs passés.
Pour chaque séquence $X = x_1, x_2, ..., x_{P_X}$, la matrice de covariance pondérée est donnée par :

$$COV_X = \frac{1}{P_X} \sum_{i=1}^{P_X} \lambda^{P_X - i} (x_i - \bar{x})(x_i - \bar{x})^T \tag{5.7}$$

où $\bar{x} = \frac{1}{P_X} \sum_{i=1}^{P_X} x_i$ est le vecteur moyenne et $0 < \lambda \leq 1$.

Le paramètre λ représente l'ordre des vecteurs dans la séquence ainsi que le poids du passé dans la représentation de la séquence.

Pour $\lambda = 1$, nous retrouvons la matrice de covariance classique, tous les vecteurs ont le même poids et l'ordre n'est pas important dans la séquence. Plus λ est proche de 0, plus les vecteurs passés ont un faible poids dans la séquence.
La distance utilisée est celle de Frobenius et l'algorithme d'apprentissage est le même que celui décrit dans la seconde approche.

5.2.4 Approche 4 : Modélisation par une matrice de covariance dynamique (M-SOM (DCOV))

Dans cette approche, le vecteur moyenne $\bar{x}(t)$ de chaque vecteur $x(t) \in R^n$ dans la séquence est obtenu en utilisant les vecteurs précédents et courant $\{x_i\}$, $(1 \leq i \leq t)$, il est calculé de façon dynamique de la manière suivante :

$$\bar{x}(t) = \frac{1}{t}\sum_{i=1}^{t} x_i \quad (t \geq 2) \tag{5.8}$$

Pour chaque séquence $X = x_1, x_2, ..., x_{P_X}$, la matrice de covariance dynamique est calculée comme suit :

$$COV_X = \frac{1}{P_X}[x_1 x_1^T + \sum_{i=2}^{P_X}(x_i - \bar{x}_i)(x_i - \bar{x}_i)^T] \tag{5.9}$$

Ce modèle représente l'ordre temporel dans la séquence en associant à chaque vecteur de la séquence son propre vecteur moyenne. Ce vecteur moyenne dépend seulement des vecteurs passés et du vecteur courant.

Cette représentation permet un calcul incrémental de la matrice de covariance. Ceci est utile lorsque les matrices de covariance sont calculées au fur et à mesure de l'arrivée des vecteurs dans une séquence (comme dans la tâche de prédiction).

Formellement, pour la séquence courante $X(t+1) = (x_1, ..., x_t, x_{t+1})$ à l'instant $t+1$, la nouvelle matrice de covariance $COV_{X(t+1)}$ peut être calculée en utilisant la précédente $COV_{X(t)}$ associée à la séquence $X(t) = (x_1, ..., x_t)$ comme suit :

$$COV_{X(t+1)} = \frac{t}{t+1}COV_{X(t)} + \frac{1}{t+1}[(x_{t+1} - \bar{x}_{t+1})(x_{t+1} - \bar{x}_{t+1})^T] \tag{5.10}$$

De même que pour l'approche précédente, la distance utilisée est celle de Frobenius et l'algorithme d'apprentissage est le même que celui décrit dans la seconde approche.

5.2.5 Approche 5 : Modélisation par un modèle auto-régressif vectoriel

Les modèles auto-régressifs vectoriels ont fait l'objet de plusieurs études en reconnaissance automatique du locuteur (Magrin-Chagnolleau *et al.*, 1996) et (Magrin-Chagnolleau, 1997). Leur utilisation a souvent été motivée par la conviction que cette approche était un moyen de modéliser

les caractéristiques dynamiques du locuteur.
Dans cette approche, pour une séquence $X = x_1, x_2, ..., x_{P_X}$, le vecteur centré est défini par $x_i^* = x_i - \bar{x}_i$ où $\bar{x}_i$ est le vecteur moyenne de la séquence $\{x_i\}$, $(1 \leq i \leq P_X)$.
Notons χ_k les matrices de covariance décalées définies par :

$$\chi_k = \frac{1}{M} \sum_{t=k+1}^{M} x_t^* . x_{t-k}^{*T} \; avec \; k = 1, ..., q \tag{5.11}$$

et T_q la matrice bloc-Toeplitz définie par :

$$T_q = \begin{pmatrix} \chi_0 & \chi_1 & \cdots & \chi_q \\ \chi_1^T & \chi_0 & \cdots & \chi_{q-1} \\ \vdots & \vdots & \ddots & \vdots \\ \chi_q^T & \chi_{q-1}^T & \cdots & \chi_0 \end{pmatrix} \tag{5.12}$$

Un modèle AR-vectoriel d'ordre q de la séquence X s'écrit classiquement :

$$\sum_{i=0}^{q} A_i . x_{t-i}^* = e_t \; avec \; A_0 = I_p \tag{5.13}$$

où $\{A_i\}$ est un ensemble de $q+1$ coefficients matriciels de prédiction linéaire et e_t est le vecteur d'erreurs de prédiction. Les coefficients matriciels $\{A_1, ..., A_q\}$ sont obtenus en résolvant l'équation vectorielle de Yule-Walker (Magrin-Chagnolleau, 1997).
Par exemple, dans un modèle AR-vectoriel d'ordre 2 (q=2), les coefficients matriciels $\{A_1, A_2\}$ sont obtenus en résolvant l'équation :

$$\begin{pmatrix} A_1 & A_2 \end{pmatrix} . \begin{pmatrix} \chi_0 & \chi_1 \\ \chi_1^T & \chi_0 \end{pmatrix} = - \begin{pmatrix} \chi_1^T & \chi_2^T \end{pmatrix} \tag{5.14}$$

Plusieurs mesures de similarité entre les modèles AR-vectoriels ont été définies dans (Magrin-Chagnolleau *et al.*, 1996) et (Magrin-Chagnolleau, 1997), nous pouvons citer les mesures symétriques qui sont définies comme suit :
Étant donné deux séquences $X = x_1, x_2, ..., x_{P_X}$ et $Y = y_1, y_2, ..., y_{P_X}$ représentées par les modèles AR-vectoriels $\chi_q = \{A, X_q\}$ et $\Upsilon_q = \{B, Y_q\}$:

$$f_{\chi_q}^{(B/A)\star} = \frac{1}{2} f_{\chi_q}^{(B/A)} + \frac{1}{2} f_{\Upsilon_q}^{(A/B)} \tag{5.15}$$

$$f_{\chi_q}^{(B/A)\diamond} = \frac{M}{M+N} f_{\chi_q}^{(B/A)} + \frac{N}{M+N} f_{\Upsilon_q}^{(A/B)} \tag{5.16}$$

$$f_{\chi_q}^{(B/A)\bullet} = \frac{N}{M+N} f_{\chi_q}^{(B/A)} + \frac{M}{M+N} f_{\Upsilon_q}^{(A/B)} \tag{5.17}$$

Où : f est choisie comme étant une combinaison des fonctions suivantes :

$a(\Gamma) = \frac{1}{p} tr(\Gamma)$ et $g(\Gamma) = [det(\Gamma)]^{\frac{1}{p}}$

avec $\Gamma_{\chi_q}^{(B/A)} = (A\chi_q A^T)^{-\frac{1}{2}}.B\chi_q B^T.(A\chi_q A^T)^{-\frac{1}{2}}$

Ces mesures nécessitent le calcul de déterminants ou d'inverses de matrices, donc un temps de calcul important. C'est pour cela que nous n'avons pas testé cette approche pour notre application[10].

5.3 Discussion

Des mesures plus adaptées pour comparer des matrices de covariances peuvent être utilisées. Ces mesures sont utilisées dans le domaine de la reconnaissance de la parole et utilisent une mesure basée sur un test de sphéricité (Bimbot et Mathan, 1993; Magrin-Chagnolleau, 1997). Le test de sphéricité constitue un test d'hypothèse d'égalité de deux matrices de covariance.
Pour deux matrices de covariance COV_X et COV_Y, le test de sphéricité constitue un test de proportionnalité de ces matrices à l'aide d'une fonction de vraisemblance qui combine deux critère : l'un sur la diagonalité de la matrice $COV_Y COV_X^{-1}$, et l'autre sur l'égalité des éléments diagonaux de cette matrice, sachant qu'elle est diagonale.
La mesure de sphéricité permet en fait de mesurer à quel point les valeurs propres d'une matrice sont toutes égale, c'est à dire à quel point cette matrice est proportionnelle à l'identité. Comme la représentation de la matrice identité dans un espace multi-dimensionnel est une sphère, la mesure de sphéricité est donc une façon de mesurer à quel point la représentation de la matrice $COV_Y COV_X^{-1}$ est sphérique, d'où son nom.

Parmi les mesures obtenues à partir du test de sphéricité, nous pouvons utiliser la distance (sd) entre les matrices de covariance $COV_X \in R^n \times R^n$ et les poids d'un neurone $W_c \in R^n \times R^n$, qui est aussi une matrice de covariance, définie comme suit :

$$sd(COV_X, W_c) = log[tr(COV_X.W_c^{-1})tr(COV_X^{-1}.W_c)] - 2log(n) \tag{5.18}$$

Cette distance peut être utilisée à la place de la distance de Frobenius pour comparer tous les modèles matriciels décrits ci-dessus.

La formule d'adaptation utilisée dans M-SOM est de la forme :

$$w_{kl}^c(t+1) = w_{kl}^c(t) - \epsilon h_{cs} \frac{\partial sd(COV_X, W_c)}{\partial w_{kl}^c} \tag{5.19}$$

où :

[10] Une parallélisation de l'algorithme d'apprentissage de ces modèles pourrait résoudre ce problème.

$COV_X = x_{ij}(1 \leq i,j \leq n)$, $COV_X^{-1} = \tilde{x}_{ij}$ $(1 \leq i,j \leq n)$,

$W_c = w_{ij}^c(1 \leq i,j \leq n)$, $W_c^{-1} = \tilde{w}_{ij}^c$ $(1 \leq i,j \leq n)$,

ϵ est le pas d'apprentissage et h_{cs} est la fonction de voisinage.

Le calcul de $\frac{\partial sd(COV_X,W_c)}{\partial w_{kl}^c}$ s'effectue comme suit :

- Si $k \neq l$:

$$\frac{\partial sd(COV_X,W_c)}{\partial w_{kl}^c} = \frac{2\tilde{x}_{kl}\sum_{i=1}^n\sum_{j=1}^n \tilde{w}_{ij}^c x_{ij} + \sum_{i=1}^n\sum_{j=1}^n w_{ij}^c \tilde{x}_{ij} * \sum_{i=1}^n\sum_{j=1}^n \frac{\partial \tilde{w}_{ij}^c}{\partial w_{kl}^c} x_{ij}}{sd(COV_X,W_c)+2log(n)}$$

- Si $k = l$:

$$\frac{\partial sd(COV_X,W_c)}{\partial w_{kl}^c} = \frac{\tilde{x}_{kl}\sum_{i=1}^n\sum_{j=1}^n \tilde{w}_{ij}^c x_{ij} + \sum_{i=1}^n\sum_{j=1}^n w_{ij}^c \tilde{x}_{ij} * \sum_{i=1}^n\sum_{j=1}^n \frac{\partial \tilde{w}_{ij}^c}{\partial w_{kl}^c} x_{ij}}{sd(COV_X,W_c)+2log(n)}$$

Le calcul de $\frac{\partial \tilde{w}_{ij}^c}{\partial w_{kl}^c}$ se fait de manière assez simple à partir de l'égalité :

$$W_c.W_c^{-1} = Id$$

$$\Rightarrow \quad (\frac{\partial}{\partial w_{kl}^c}W_c)W_c^{-1} + W_c(\frac{\partial}{\partial w_{kl}^c}W_c^{-1}) = 0$$

Nous avons alors :

$$\frac{\partial}{\partial w_{kl}^c}W_c^{-1} = -W_c^{-1}(\frac{\partial}{\partial w_{kl}^c}W_c)W_c^{-1}$$

si $k \neq l$, la matrice $\frac{\partial}{\partial w_{kl}^c}W_c$ est une matrice dont tous les éléments sont nuls sauf celui de la k^{eme} ligne et l^{eme} colonne et celui de la l^{eme} ligne et k^{eme} colonne qui valent 1.

si $k = l$, la matrice $\frac{\partial}{\partial w_{kl}^c}W_c$ est une matrice dont tous les éléments sont nuls sauf celui de la k^{eme} ligne et k^{eme} colonne qui vaut 1.

Les mesures basées sur le test de sphéricité sont plus coûteuses en temps que celle de Frobenius car elles nécessitent le calcul d'inverses de matrices ou de déterminants de matrices (c'est pour cela que nous ne les avons pas utilisées pour notre application[11].), mais nous pensons qu'elles peuvent donner de meilleurs résultats.

Dans le paragraphe suivant, le modèle M-SOM-ART qui combine les deux modèles SOM-ART et M-SOM est présenté. La modélisation utilisée est celle de la matrice de covariance dynamique

[11] Une parallélisation de l'algorithme d'apprentissage de ces modèles pourrait résoudre ce problème.

(les autres modèles peuvent aussi être utilisés).

5.4 Incorporation de l'incrémentalité dans M-SOM

Nous proposons une nouvelle approche pour la classification et le classement de séquences utilisant une carte auto-organisatrice évolutive. Ce modèle baptisé « M-SOM-ART » (voir la figure 5.4) combine les deux cartes SOM-ART et M-SOM (Zehraoui et Bennani, 2004a). Dans cette approche, nous modélisons la séquence par une matrice de covariance dynamique. Tous les composants de la séquence sont pris en compte. La position et la forme du nuage de points qui représentent la séquence sont modélisés à l'entrée du réseau SOM. Le problème de la longueur variable ne se pose pas puisque toutes les entrées sont de même dimension. De plus la carte M-SOM est intégrée dans un paradigme ART, ceci permet de la doter des propriétés de plasticité et de stabilité.

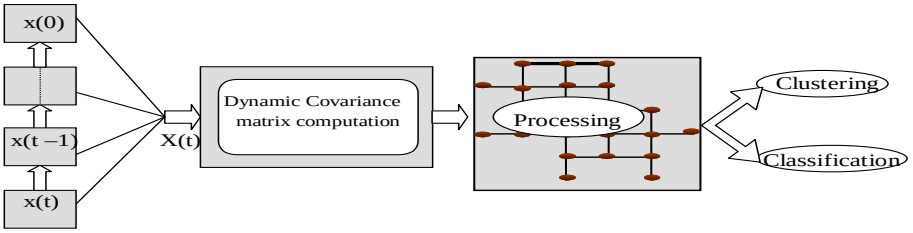


FIG. *5.4: Architecture de M-SOM-ART.*

Une séquence X est définie comme un ensemble fini ordonné de vecteurs $x_i \in R^n$ $(1 \leq i \leq p)$. Dans cette approche, l'entrée de la séquence est modélisée par sa matrice de covariance dynamique $COV_X \in R^n \times R^n$ définie comme suit :

$$COV_X = \frac{1}{p}[x_1 x_1^T + \sum_{i=2}^{p}(x_i - \bar{x}_i)(x_i - \bar{x}_i)^T] \tag{5.20}$$

où $\bar{x}(t) = \frac{1}{t}\sum_{i=1}^{t} x_i \quad (t \geq 2)$ est la moyenne dynamique associée à $x_t \in R^n$ dans la séquence et calculée en utilisant les vecteurs précédents et courant $\{x_i\}$, $(1 \leq i \leq t)$. La distance entre une matrice de covariance $COV_X = (x_{ij})$, $1 \leq i, j \leq n$ et les poids du neurone $W_c = (w_{ij}^c)$, $1 \leq i, j \leq n$ est la distance matricielle de Frobenius (fd) donnée par :

$$\begin{aligned} fd(COV_X, W_c) &= [tr(COV_X - W_c)^T(COV_X \\ &-W_c)]^{1/2} = [\sum_i \sum_j (x_{ij} - w_{ij}^c)^2]^{1/2} \end{aligned} \tag{5.21}$$

où $tr(M)$ est la trace de la matrice M et x^T est la transposée du vecteur x.
La carte M-SOM-ART est d'abord initialisée par un neurone, puis de nouveaux neurones sont

ajoutés à la carte suivant le paradigme SART qui utilise le test de vigilance pour sélectionner le vainqueur. Si la contrainte de vigilance est satisfaite, la carte est mise à jour ; sinon un neurone est ajouté à la carte. Le neurone du périmètre de la carte le plus proche de l'entrée est déterminé et un nouveau neurone est ajouté dans son voisinage. Les connexions correspondantes sont aussi ajoutées. Comme dans les modèles précédents, des étiquettes sont associées aux neurones de la carte durant la dernière phase d'apprentissage. La carte SOM dans ce modèle effectue la classification et le classement de séquences. L'algorithme d'apprentissage est décrit dans l'algorithme 6.

Algorithme 6 Algorithme d'apprentissage de M-SOM-ART.

Initialiser l'ensemble A des neurones à un neurone c_1.

Initialiser le paramètre temps t : $t = 0$.

Initialiser l'ensemble de connexions $C \subset A \times A$ à l'ensemble vide : $C = \emptyset$.

Tant que $(t \leq t_{max})$ **faire**

1. Pour toute séquence d'entrée $X = (x(i))$ $(1 \leq i \leq p)$, $x(i) \in R^n$
   - Modéliser l'entrée X par sa matrice de covariance dynamique :

$$COV_X = \frac{1}{p}[x(1)x(1)^T + \sum_{i=2}^{p}(x(i) - \bar{x}(i))(x(i) - \bar{x}(i))^T]$$

   Initialiser le vecteur poids W_{c1} au premier neurone c_1 de la matrice de covariance dynamique de la première séquence d'entrée.
   - Déterminer le neurone gagnant $s(X) \in A$ par :

$$s(X) = \arg\min_{c \in A} df(COV_X, W_r)$$

   où df est la distance de Frobenius.
   - Soumettre le neurone gagnant $s(X)$ au test de vigilance :
   - Si $(\frac{1}{1+df(cov_x, s(X))} \geq \rho$: $\rho \in [0,1]$ est le paramètre de vigilance, adapter chaque neurone c_r comme suit :

$$\Delta W_{c_r} = \epsilon(t) h_{rs}[COV_X - W_{c_r}]$$

   $\epsilon(t)$ est le pas d'apprentissage, σ est la fonction de déviation Gaussienne, σ_i et h_{rs} est la fonction de voisinage
   - sinon ajouter un nouveau neurone c_r dans la voisinage du neurone du périmètre de la carte le plus proche de l'entrée, initialiser le vecteur poids W_{c_r} par : $W_{c_r} = COV_X$.
2. Si $(t = t_{max})$ étiqueter chaque neurone en utilisant un vote majoritaire sur les étiquettes des entrées qui ont activé ce neurone.
3. Incrémenter le paramètre temps t : $t = t + 1$

Le tableau 5.3 montre les propriétés de M-SOM-ART comparées aux propriétés d'autres

	M-SOM-ART	SART	SOM	SOM incrémentales
Stabilité	+	+	-	-
Plasticité	+	+	-	+
Classement	+	-	+	+
Classification	+	+	+	+
Visualisation	+	-	+	+
traitement des séquences	+	-	+	-

TAB. *5.2: Comparaison des propriétés de M-SOM-ART et d'autres réseaux de neurones.*

cartes SOM et des modèles SART. Le symbole '+' représente la présence de la propriété dans le modèle ou dans certaines de ses variantes et '-' représente l'absence de la propriété.

Dans le paragraphe suivant, nous présentons les différentes expérimentations effectuées pour valider les modèles présentés.

5.5 Validation des approches proposées

5.5.1 Cadre applicatif

Pour valider les modèles que nous avons proposés, nous avons effectué des expérimentations sur les fichiers log d'un site de commerce électronique. Plus précisément, le comportement de chaque utilisateur est décrit par des informations sur la succession de pages qu'il a parcourues dans le temps. Cette succession de pages représente l'aspect temporel des données. De plus, le site de *e-commerce* est consulté chaque jour par des milliers de personnes, par conséquent, nous avons un grand volume de données qui sont dynamiques puisqu'elles peuvent constamment changer. Comme notre travail est réalisé pour pouvoir effectuer des actions durant la navigation de l'internaute, le traitement des données doit être effectué avant qu'il ne quitte le site, ce qui impose des contraintes temps réel. Dans notre application, nous ne possédons pas de connaissances *a priori* du domaine et les données contiennent du bruit.
Les différents traitements des données cités dans le paragraphe §1.2 ont été effectués avant d'utiliser les modèles proposés (voir la figure 5.5).

5.5.2 Principe général

Les navigations représentent des successions de ces vecteurs (correspondant aux successions de pages) de longueur variable. Une classification des navigations suivant les comportements des internautes ainsi qu'un classement suivant l'une des deux classes {acheteur, non acheteur} sont effectués en utilisant les différents modèles proposés.

Dans le modèle SOM-ART, un classement est effectué pour chaque état d'une navigation[12] (fenêtre temporelle de longueur 1). Dans les autres modèles (M-SOM et M-SOM-ART), les classements des navigations sont obtenus directement par leurs sorties (voir la figure 5.5).
Pour évaluer les modèles proposés, nous avons utilisé les mesures suivantes :

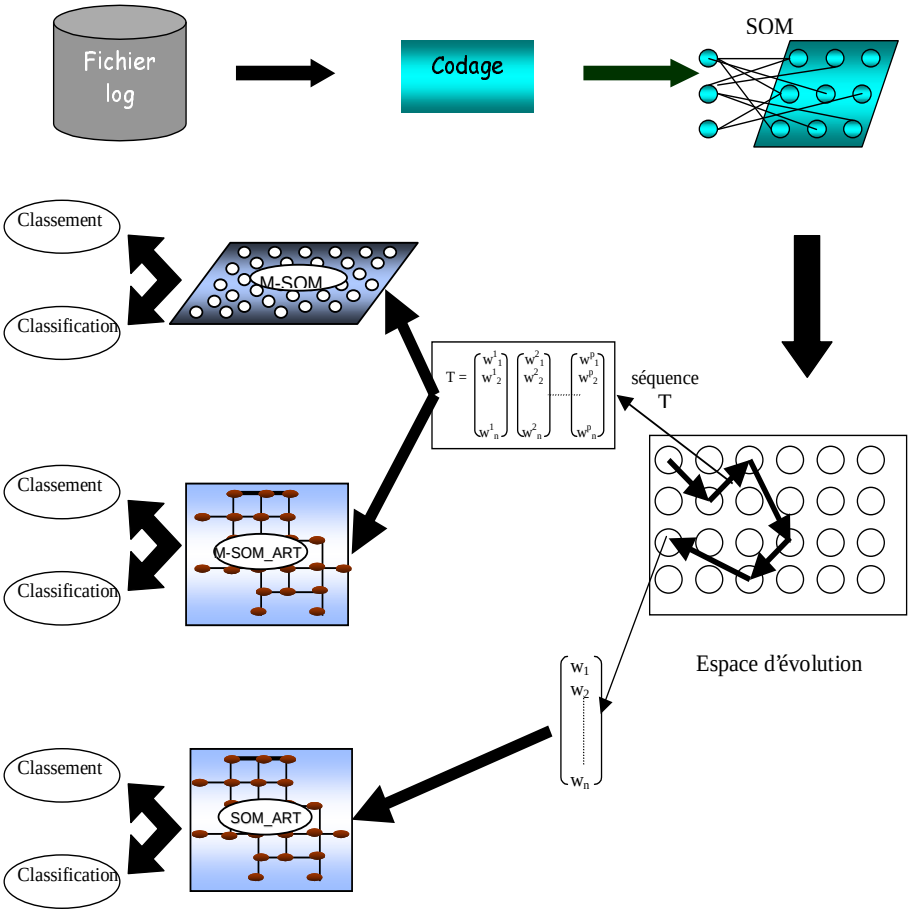


FIG. 5.5: *Les différents traitements effectués sur les données.*

– *Résultat global :* représente le taux de classements corrects sur le nombre de tous les classements ($\frac{nombre\_classements\_corrects}{nombre\_classements}$). Ce résultat est donné sous forme d'un intervalle de confiance à 95% (Bennani, 1992).

– *Matrice de confusion :*

$$\begin{pmatrix} A & \overline{N} \\ \overline{A} & N \end{pmatrix}$$

[12]Le classement d'une navigation entière peut être obtenu par vote majoritaire sur les classements des vecteurs qui la constituent.

où :

A = nombre de classements corrects des acheteurs.

$\overline{A}$ =nombre de classements incorrects des non acheteurs.

$\overline{N}$ = nombre de classements incorrects des acheteurs.

N =nombre de classements corrects des non acheteurs.

- *Sensibilité :* représente le taux de bons classements de la classe « acheteurs » ($\frac{A}{A+\overline{N}}$)
- *Spécificité :* représente le taux de classements corrects de la classe « non acheteurs » ($\frac{N}{\overline{A}+N}$)
- *L'espace ROC (Receiver Operating Characteristics) :* est une représentation graphique du compromis entre les taux de classements corrects des acheteurs et le taux de classements incorrects des non acheteurs. L'axe des abscisses représente 1- spécificité et l'axe des ordonnées représente la stabilité. Chaque classifieur est représenté par un point de coordonnées (1 - spécificité, sensibilité). Le meilleur classifieur est celui qui est proche du nord ouest (0,1).
- *Le nombre de neurones de la couche de sortie :* Ce nombre a une influence directe sur le temps de traitement des données. Plus le nombre de neurones est petit, plus le temps de traitement est court.

5.5.3 Validation du modèle SOM-ART

5.5.3.1 Incrémentalité en structure

	SOM-ART [Zehraoui & Bennani, 2004]				GG [Fritzke,1995]	SOM [Kohonen, 1995]
Seuil de vigilance	0.9	0.95	0.97	0.98	--	--
Nombre de neurones	155	200	302	3 57	357	1026
Résultat global	[36.70%, 8.11%]	[56,10%, 7.55%]	[62.98%,64.38%]	[84.54%, 85.58%]	[84.56%,85.60%]	[84.55%, 85.59%]
Matrice de confusion	3945 11068 226 2807	2788 6408 1383 7467	2447 4828 1724 9047	1735 265 2436 13610	1733 261 2438 13614	1730 259 2441 13616
Sensibilité	**0.948**	**0.668**	**0.586**	**0.416**	**0.415**	**0.414**
Spécificité	**0.202**	**0.538**	**0.652**	**0.981**	**0.981**	**0.981**

FIG. *5.6: Comparaison des résultats obtenus pour le classement des états des séquences en utilisant SOM, GG et SOM-ART.*

Nous avons d'abord comparé les résultats obtenus par SOM-ART en faisant varier le seuil de

vigilance et ceux obtenus par SOM et par le modèle GG en utilisant une base d'apprentissage qui contient 7000 navigations (41030 états) et une base de test qui contient 3000 navigations (18046 états). Les navigations sont de longueurs variables. Ces tests permettent de montrer l'incrémentalité de SOM-ART en structure.

Les tests montrent que le nombre de neurones augmente avec l'augmentation du seuil de vigilance dans le réseau SOM-ART ainsi que le résultat global. Celui-ci se stabilise et devient égal à celui de SOM pour un seuil de vigilance égal à 0.95 (voir le tableau de la figure 5.6).

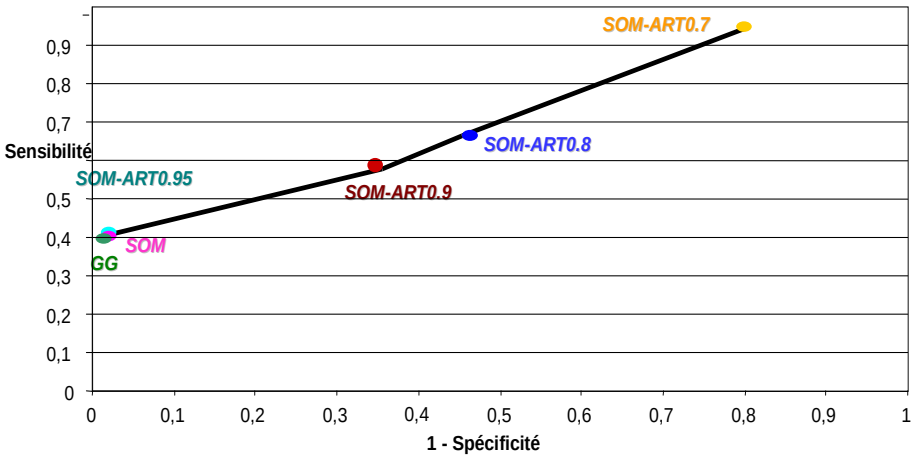


FIG. 5.7: *Espace ROC : Comparaison de résultats obtenus pour le classement des états des séquences en utilisant SOM, GG et SOM-ART.*

La sensibilité diminue avec l'augmentation du seuil de vigilance tandis que la spécificité augmente. Dans l'espace ROC, nous remarquons que la carte SOM, le modèle GG ainsi que SOM-ART pour un seuil de vigilance égal à 0.95 sont les plus proches du nord ouest (voir la figure 5.8), donc fournissent les meilleurs résultats de classement. De plus, le nombre de neurones de SOM-ART est inférieur à celui de SOM, donc le temps mis par SOM-ART et GG pour traiter les données est inférieur à celui mis par SOM.

5.5.3.2 Incrémentalité en données (plasticité/stabilité)

Pour montrer l'incrémentalité de SOM-ART en données (plasticité et stabilité), nous avons partitionné la base de données entière H (base d'apprentissage + base de test) en 5 petites bases : A, B, C, D et E. Ces bases sont utilisées successivement dans le processus d'apprentissage des trois modèles (SOM, GG et SOM-ART). Après chaque phase d'apprentissage utilisant l'une des cinq petites bases X ($X \in \{A, B, C, D, E\}$), la phase de test est effectuée en utilisant la base (H-X).

	Bases	Résultat global	Sensibilité	Spécificité
SOM [Kohonen,1995]	H-A	[85.55, 86.18]	0.437	0.979
	H-B	[85.59, 86.22]	0.438	0.982
	H-C	[79.08, 79.80]	0.146	0.981
	H-D	[80.06, 80.78]	0.147	0.982
	H-E	[80.16, 80.88]	0.151	0.981
SOM-ART [Zehraoui & Bennani,2004]	H-A	[85.56, 86.19]	0.436	0.978
	H-B	[85.58, 86.21]	0.438	0.981
	H-C	[85.74, 86.37]	0.439	0.981
	H-D	[86.30, 86.91]	0.442	0.982
	H-E	[86.59, 87.20]	0.452	0.981
GG [Fritzke,1995]	H-A	[85.56, 86.18]	0.439	0.982
	H-B	[84.98, 85.62]	0.375	0.989
	H-C	[85.75, 86.38]	0.441	0.981
	H-D	[86.23, 86.85]	0.438	0.980
	H-E	[85.70, 86.33]	0.468	0.965

FIG. *5.8: Comparaison des résultats obtenus après le partitionnement de la base de données en utilisant SOM, SOM-ART et GG.*

La table de la figure 5.8 et la figure 5.9 montrent que les résultats obtenus en utilisant SOM-ART sont améliorés après chaque phase d'apprentissage effectué en utilisant l'une des cinq petites bases contenues dans la base de test. Ceci n'est pas le cas pour SOM et GG.
Le tableau 5.3 montre les résultats obtenus en utilisant la base d'apprentissage entière H (A + B + C + D + E) et la dernière base de test H-E pour les trois cartes. En comparant ces résultats à ceux obtenus après des apprentissages successifs sur les cinq bases (A, B, C, D et E) (voir les dernières lignes de chaque modèle illustré dans le tableau de la figure 5.8), nous remarquons que les résultats de SOM-ART sont proches de ceux obtenus en effectuant des apprentissages successifs sur les petites bases. Les résultats obtenus par GG et SOM en utilisant les petites bases sont moins bons que ceux obtenus en utilisant la base d'apprentissage entière.

	SOM-ART [Zehraoui & Bennani,2004]	SOM [Kohonen,1995]	GG [Fritzke,1995]
Résultat global	[86.58%, 87.19%]	[86.55%, 87.16%]	[86.60%, 87.22%]
Sensibilité	0.451	0.446	0.449
Spécificité	0.980	0.981	0.982

TAB. *5.3: Comparaison des résultats de classement de SOM-ART, SOM et GG en utilisant la base entière.*

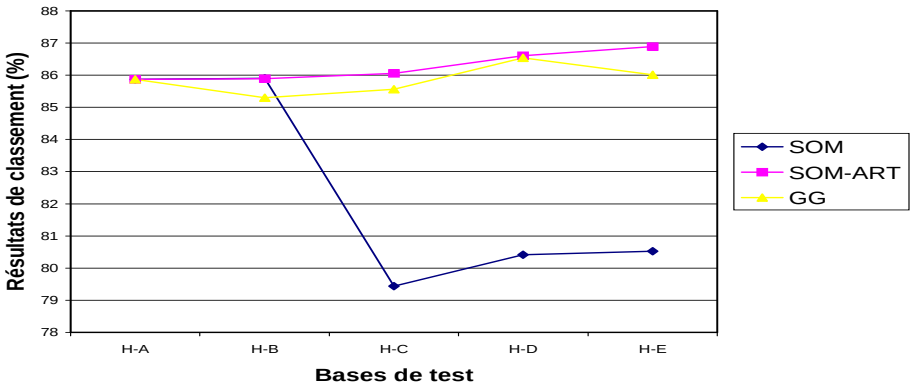


FIG. *5.9: Comparaison du résultat global de classement obtenu après le partitionnement de la base de données en utilisant SOM, SOM-ART et GG.*

Les tests que nous avons effectués montrent qu'avec SOM-ART, on a pu obtenir les mêmes résultats de classement tout en ayant moins de neurones, de plus notre approche possède les propriétés de stabilité et de plasticité.

5.5.4 Validation des modèles M-SOM

Pour évaluer les différentes approches : « SOM temporelle utilisant des vecteurs propres (VectSOM) », « M-SOM utilisant la matrice de covariance (M-SOM (CM)) », « M-SOM utilisant la matrice de covariance pondérée (M-SOM (WCM)) » et « M-SOM utilisant la matrice de covariance dynamique (M-SOM (DCM)) », nous avons comparé leurs résultats de classement à ceux d'autres modèles de SOM. Ces modèles sont « SOTPAR (SOTPAR) » qui utilise une représentation interne de l'information temporelle, « la carte de Kohonen SOM (SOM) », qui ne prend pas en compte l'aspect temporel des données et « SOM avec délai exponentiel pondéré (SOMTemp) » qui utilise une représentation externe de l'information temporelle.
Dans les approches que nous avons proposées, chaque séquence est représentée par un vecteur ou une matrice, le classement de la séquence est obtenu directement par la carte. Dans les autres approches, un classement est effectué pour chaque vecteur de la séquence. Le classement global de la séquence est obtenu par vote majoritaire sur les résultats fournis lors du classement des vecteurs constituant la séquence[13].
Dans les expérimentations, nous avons utilisé deux bases de données : une base d'apprentissage, qui contient 7000 navigations et une base de test qui contient 3000 navigations. Dans les deux bases, il y a moins de 8% d'acheteurs.

[13]Dans le cadre de notre application "e-commerce", si nous obtenons le même nombre d'acheteurs et de non acheteurs dans une séquence, nous choisissons la classe la plus rare (les acheteurs représentent moins de 8% des séquences dans la base de données)

	Résultat global	Matrice de confusion	Sensibilité	Spécificité
SOM [Kohonen,1995]	[94.54%, 96.05%]	$\begin{pmatrix} 109 & 28 \\ 111 & 2752 \end{pmatrix}$	0.495	0.989
SOTPAR [Euliano,1998]	[95.48%, 96.85%]	$\begin{pmatrix} 114 & 07 \\ 106 & 2773 \end{pmatrix}$	0.518	0.997
SOMTemp [Kangas,1991]	[97.27%, 98.31%]	$\begin{pmatrix} 160 & 04 \\ 60 & 2776 \end{pmatrix}$	0.727	0.998
VectSOM [Zehraoui & Bennani,2004]	[99%, 99.58%]	$\begin{pmatrix} 211 & 10 \\ 09 & 2770 \end{pmatrix}$	0.959	0.996
M-SOM [Zehraoui & Bennani,2004]	[99.09%, 99.64%]	$\begin{pmatrix} 214 & 11 \\ 06 & 2769 \end{pmatrix}$	0.972	0.996
M-SOM (WCM) [Zehraoui & Bennani,2004]	[99.34%, 99.79%]	$\begin{pmatrix} 216 & 07 \\ 04 & 2773 \end{pmatrix}$	0.981	0.997
M-SOM (DCM) [Zehraoui & Bennani,2004]	[99.30%, 99.77%]	$\begin{pmatrix} 214 & 06 \\ 06 & 2774 \end{pmatrix}$	0.972	0.998

FIG. *5.10: Résultats de classement de séquences.*

Les résultats expérimentaux (voir le tableau de la figure 5.10) montrent que notre approche donne de meilleurs résultats globaux et de meilleures sensibilités que SOM, SOTPAR et SOM-Temp.
A part SOM, qui n'est pas adaptée pour le traitement des entrées temporelles, les résultats de spécificité dans toutes les approches sont proches. Toutes les approches temporelles donnent de bons résultats pour le classement des séquences fréquentes (les non acheteurs représentent plus de 92% des séquences dans la base de données), mais les approches que nous avons proposées donnent de meilleurs résultats pour le classement des séquences rares (les séquences d'acheteurs).
L'espace ROC (voir la figure 5.11) montre que les modèles que nous proposons donnent de meilleurs résultats de classement que les autres approches (les classifieurs associés sont proches du nord ouest : les distances peuvent être calculées facilement).
Notons que les deux dernières approches (M-SOM (WCM) et M-SOM (DCM)) donnent de meilleurs résultats que les deux premières. La prise en compte de la dynamique temporelle dans la matrice de covariance permet d'améliorer les résultats de classement.

5.5.5 Validation du modèle M-SOM-ART

Nous avons d'abord comparé les résultats de classement obtenus par M-SOM-ART à ceux obtenus par d'autres modèles SOM. Ces modèles sont SOTPAR (SOTPAR) qui utilise une représentation interne des informations temporelles, la carte de Kohonen (SOM), qui ne prend pas en compte l'aspect temporel des données et SOM avec décalage exponentiel (SOMTemp) qui utilise une représentation externe de l'information temporelle.

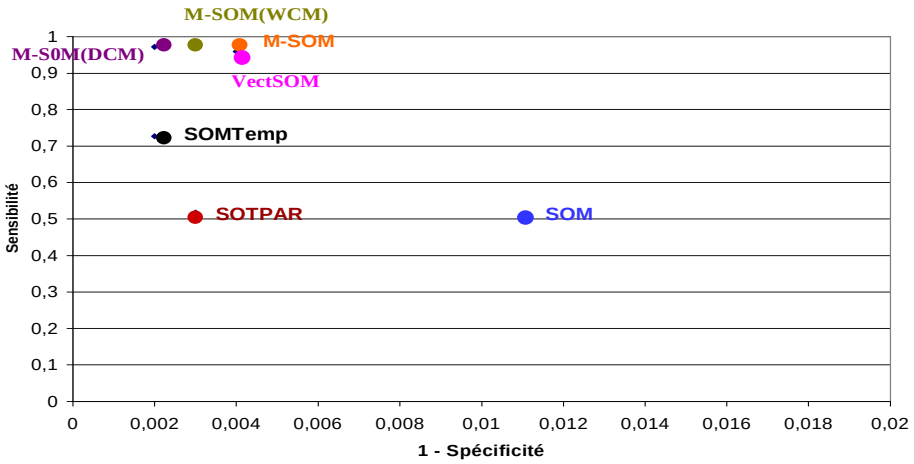


FIG. *5.11: Espace ROC.*

Dans notre approche, chaque séquence est représentée par un vecteur ou une matrice, le classement de la séquence est obtenu directement par la carte. Dans les autres approches, un classement est effectué pour chaque vecteur de la séquence. Le classement global de la séquence est obtenu par vote majoritaire sur les résultats fournis lors du classement des vecteurs constituant la séquence. Nous avons comparé les résultats obtenus par les différentes approches pour le classement des navigations. Nous avons utilisé les mêmes bases que précédemment : une base d'apprentissage qui contient 7000 navigations et une base de test qui contient 3000 navigations. Dans ces bases de données, il y a moins de 8% d'acheteurs.

	Résultat global	Matrice de confusion	Sensibilité	Spécificité
SOM [Kohonen, 1995]	**[94.54%, 96.05%]**	$\begin{pmatrix} 109 & 28 \\ 111 & 2752 \end{pmatrix}$	**0.495**	**0.989**
SOTPAR [Euliano, 1998]	**[95.48%, 96.85%]**	$\begin{pmatrix} 114 & 07 \\ 106 & 2773 \end{pmatrix}$	**0.518**	**0.997**
SOMTemp [Kangas, 1991]	**[97.27%, 98.31%]**	$\begin{pmatrix} 160 & 04 \\ 60 & 2776 \end{pmatrix}$	**0.727**	**0.998**
M-SOM-ART [Zehraoui & Bennani, 2004]	**[99.26%, 99.75%]**	$\begin{pmatrix} 215 & 08 \\ 05 & 2772 \end{pmatrix}$	**0.977**	**0.997**

FIG. *5.12: Résultats de classement des séquences.*

Les résultats des premières expérimentations (voir le tableau de la figure 5.12) montre que notre approche donne les meilleurs résultats globaux et la meilleure sensibilité que SOM, SOTPAR et SOMTemp.

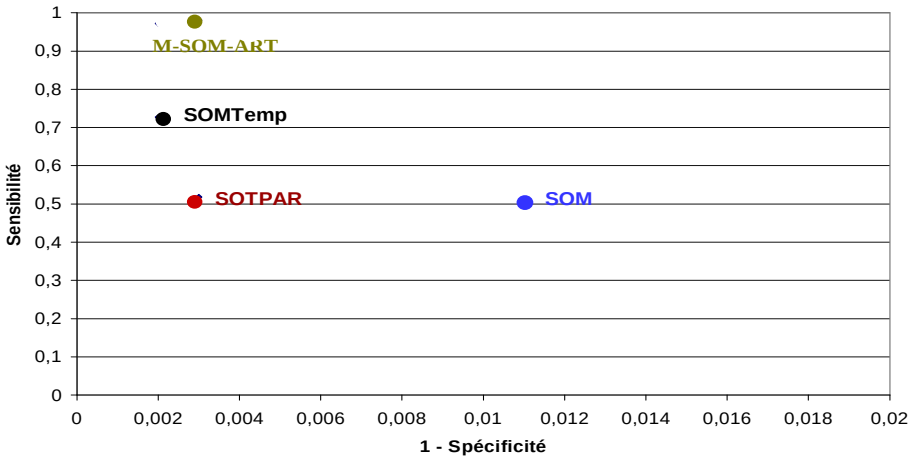


FIG. *5.13: Espace ROC : Comparaison des résultats de classement des séquences.*

A part SOM qui n'est pas adaptée pour le traitement des séquences, les résultats de spécificité sont proches dans toutes les autres approches. Toutes les approches temporelles donnent de bons résultats pour le classement de la classe la plus fréquente (les séquences des non acheteurs représentent plus de 92% de l'ensemble des séquences dans les bases de données), mais M-SOM-ART donne de meilleurs résultats pour le classement des séquences rares (les séquences acheteurs) avec une structure optimisée.

	M-SOM-ART [Zehraoui & Bennani,2004]				M-SOM [Zehraoui & Bennani,2004]
Seuil de vigilance	0.85	0.9	0.95	0.97	--
Nombre de neurones	42	114	154	289	416
Résultat global	[91,68%, 93,55%]	[94,51%, 96,03%]	[98,03%, 98,90%]	[99.26%, 99.75%]	[99.30%, 99.77%]
Matrice de confusion	00 00 220 2780	206 126 14 2654	201 25 19 2755	215 08 05 2772	214 06 06 2774
Sensibilité	**0**	**0.936**	**0.913**	**0.977**	**0.972**
Spécificité	**1**	**0.954**	**0.991**	**0.997**	**0.998**

FIG. *5.14: Comparaison des résultats de M-SOM-ART et M-SOM.*

L'espace ROC (voir la figure 5.13) montre que M-SOM-ART donne les meilleurs résultats de classement que les autres approches (leurs classifieurs sont les plus proches du nord ouest).

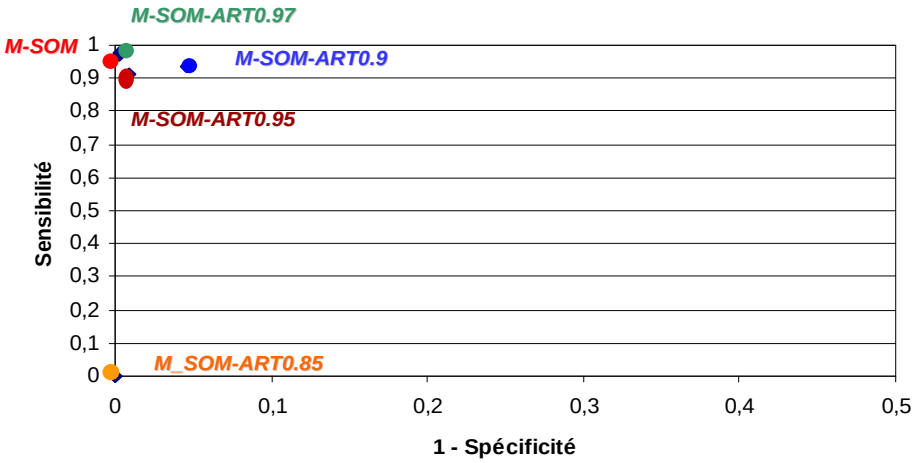


FIG. 5.15: *Espace ROC : Comparaison des résultats de M-SOM-ART et M-SOM.*

Pour évaluer la plasticité et la stabilité de notre modèle, nous l'avons comparé au modèle M-SOM, qui utilise la même modélisation de séquences que M-SOM-ART sans l'utilisation du paradigme ART.

	Bases	Résultat global	Sensibilité	Spécificité
M-SOM [Zehraoui & Bennani,2004]	H-A	[99.08%, 99.45%]	0.93	0.9981
	H-B	[96.62%, 97.36%]	0.785	0.985
	H-C	[98.1%, 98.65%]	0.956	0.986
	H-D	[98.44%, 98.93%]	0.874	0.996
	H-E	[98.15%, 98.7%]	0.967	0.986
M-SOM-ART [Zehraoui & Bennani,2004]	H-A	[99.22%, 99.55%]	0.935	0.999
	H-B	[99.29%, 99.61%]	0.972	0.9966
	H-C	[99.51%, 99.76%]	0.977	0.998
	H-D	[99.6%, 99.83%]	0.988	0.998
	H-E	[99.64%, 99.86%]	0.985	0.9987

FIG. 5.16: *Résultats obtenus en utilisant M-SOM-ART et M-SOM après la partition de la base d'apprentissage.*

5.5.5.1 Incrémentalité en structure

Dans les deuxièmes expérimentations, nous avons comparé les résultats de M-SOM-ART (modèle utilisant la matrice de covariance dynamique) pour les différentes valeurs du paramètre de vigilance à ceux obtenus par M-SOM.
La figure 5.14 montre que l'incrémentation du seuil de vigilance produit l'augmentation du nombre de neurones et du résultat global. Nous obtenons des résultats proches de ceux obte-

nus par M-SOM pour la valeur 0.97 du seuil de vigilance avec moins de neurones que M-SOM. Ceci veut dire qu'on peut avoir des résultats proches de ceux de M-SOM en moins de temps (ceci est important pour satisfaire les contraintes temps réel). L'espace ROC (voir la figure 5.15) montre que les résultats obtenus en utilisant M-SOM et M-SOM-ART pour la valeur 0.97 du seuil de vigilance sont les meilleurs car les deux classifieurs sont les plus proches du nord ouest.

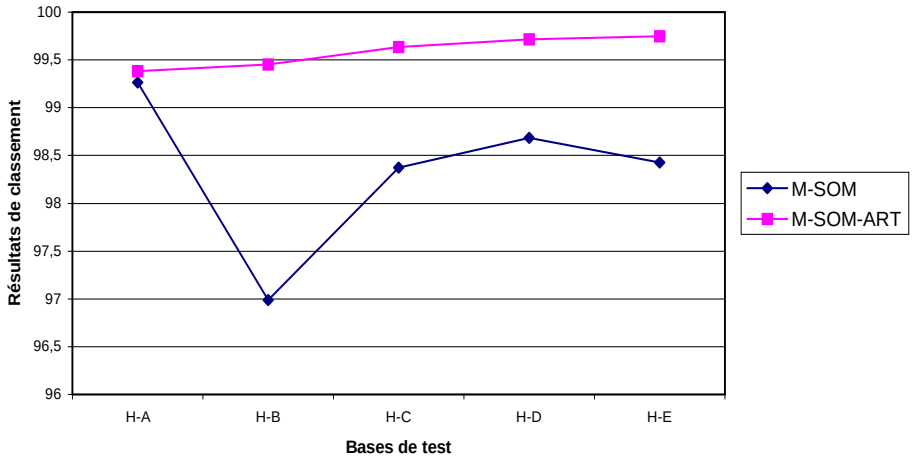


FIG. *5.17: Comparaison du résultat global de classement de M-SOM-ART et de M-SOM après la partition de la base d'apprentissage.*

5.5.5.2 Incrémentalité en données (plasticité/stabilité)

Dans les troisièmes expérimentations, nous avons partitionné la base de données entière H (base d'apprentissage + base de test) en 5 petites bases : A, B, C, D et E. Ces bases sont utilisées successivement dans le processus d'apprentissage des deux modèles (M-SOM et M-SOM-ART). Après chaque phase d'apprentissage utilisant l'une des cinq petites bases X ($X \in \{A, B, C, D, E\}$), la phase de test est effectuée en utilisant la base (H-X). La table de la figure 5.16 et la figure 5.17 montrent que les résultats obtenus en utilisant M-SOM-ART sont améliorés après chaque phase d'apprentissage effectué en utilisant l'une des cinq petites bases contenue dans la base de test. Ceci n'est pas le cas pour M-SOM.
Le tableau 5.4 montre les résultats obtenus en utilisant la base d'apprentissage entière H (A + B + C + D + E) et la dernière base de test H-E pour les trois cartes. Les résultats de M-SOM-ART sont proches de ceux obtenus en effectuant des apprentissages successifs sur les petites bases contenues dans la base de test. Les résultats obtenus par SOM en utilisant les petites bases sont moins bons que ceux obtenus en utilisant la base d'apprentissage entière. Ceci montre que M-SOM-ART est capable d'acquerir de nouvelles connaissances sans oublier les anciennes.

	Résultat global	Matrice de confusion	Sensibilité	Spécificité
M-SOM-ART [Zehraoui & Bennani,2004]	[99.42%,99.70%]	$\begin{pmatrix} 585 & 8 \\ 25 & 7382 \end{pmatrix}$	0.959	0.9989
M-SOM [Zehraoui & Bennani,2004]	[99.46%,99.74%]	$\begin{pmatrix} 593 & 13 \\ 17 & 7377 \end{pmatrix}$	0.972	0.998

TAB. *5.4: Comparaison des résultats obtenus par M-SOM et M-SOM-ART après la partition de la base initiale.*

5.6 Conclusion

Nous avons conçu et mis en œuvre plusieurs nouveaux modèles de cartes auto-organisatrices qui traitent des séquences temporelles et/ou qui sont capables d'acquérir de nouvelles connaissances sans oublier les anciennes déjà acquises. Ces modèles sont les suivants :

- Une nouvelle carte auto-organisatrice évolutive «SOM-ART» qui possède les propriétés de plasticité et de stabilité en utilisant le paradigme de la théorie de résonance adaptative.

- Une nouvelle classe de cartes SOM temporelles «M-SOM», qui permettent de modéliser les séquences en utilisant les matrices de covariance. Les premiers modèles, inspirés du domaine du traitement du signal, ne prennent pas en compte l'ordre temporel dans la séquence. Nous avons alors proposé de nouveaux modèles où cet ordre temporel est pris en compte dans la matrice de covariance.
  Dans les approches où les entrées sont modélisées par des matrices de covariance, nous avons proposé plusieurs mesures de similarité entraînant des modifications de la fonctions de distance et de la formule d'adaptation dans l'algorithme d'apprentissage de SOM.

- Le modèle «M-SOM-ART» qui combine les modèles précédemment cités. Il traite les séquences en modélisant les entrées par des matrices de covariance dynamiques et possède les propriétés de plasticité et de stabilité en se basant sur le paradigme ART.

Il reste de nombreuses directions de recherche à explorer, nous pensons tout particulièrement aux points suivants :

- Le modèle auto-régressif (AR) (Magrin-Chagnolleau *et al.*, 1996) nous parait très intéressant puisqu'il prend en compte l'ordre des vecteurs dans les séquences par l'utilisation des matrices décalées. Cette approche est considérée comme une manière efficace pour extraire la dynamique des caractéristiques du locuteur. Nous avons commencé à travailler sur ce modèle en utilisant la distance de Frobenius décrite dans le paragraphe 5.2.2, mais nous n'avons pas obtenu de bons résultats pour notre application. Les différentes distances citées dans le paragraphe 5.2.5 seront testées dans plusieurs applications et leurs résultats seront comparés à d'autres approches.

- Les modèles de cartes auto-organisatrices que nous avons proposées effectuent des classement et des classifications de données. Nous nous intéressons à utiliser ces modèles pour effectuer des prédictions. Pour ceci, plusieurs extensions de la carte SOM proposées dans la littérature peuvent être utilisées, parmi lesquelles, nous pouvons citer : la carte SOM étendue (ESOM : Extended SOM) (Ritter et Schulten, 1986), la carte SOM à sortie supervisée (SSOM : Supervised SOM) (Ritter et Schulten, 1988) et la carte SOM utilisant des fonctions à base radiales (RBF) (Park et Sandberg, 1991).

- Nous nous sommes intéressés dans notre travail particulièrement à l'évaluation des résultats de classements obtenus en utilisant les cartes auto-organisatrice que nous avons proposées puisque c'est le but de notre application. Les résultats de classification ainsi que ceux de visualisation seront étudiés. La propriété de visualisation, qui est l'une des propriétés les plus intéressantes de la carte SOM.
- Les approches connexionnistes et hybrides développées dans cette thèse ont été validées sur un problème réel qui est le traitement de séquences de navigation sur internet. Les approches proposées sont générales et peuvent traiter des données présentées sous forme de séquences multi-variées. Nos contributions sont alors facilement transposables à d'autres problèmes dans d'autres domaines (séquences d'images, séquences biologiques, données spatio-temporelles, données comportementales, etc.)

Comme le modèle M-SOM-ART possède les propriétés requises pour effectuer l'hybridation avec le RàPC, il est utilisé dans le système pour l'indexation de la base de cas et dans la phase de réutilisation ainsi que pour résoudre certains problèmes sans l'utilisation du raisonnement à partir de cas.

Troisième partie

Hybridation du RàPC et des Réseaux connexionnistes

Chapitre 6

Systèmes hybrides : État de l'art

Dans ce chapitre, nous présentons d'abord les systèmes hybrides neuro-symboliques tels qu'ils sont décrits dans la littérature, ensuite, nous décrivons les systèmes hybrides neuro-RàPC qui combinent le raisonnement à partir de cas avec les réseaux de neurones.

6.1 Systèmes hybrides neuro-symboliques

Le but de la conception des systèmes hybrides neuro-symboliques est de tirer parti des points forts de différents paradigmes afin d'obtenir de meilleures performances dans la résolution des problèmes. Ces systèmes sont composés d'au moins deux modules différents. Par exemple, les réseaux de neurones permettent de traiter de gros volumes de données de manière efficace, ce qui n'est généralement pas le cas pour les systèmes symboliques. Par contre, ceux-ci permettent de justifier les résultats qu'ils fournissent, ce qui n'est pas toujours possible lors de l'utilisation des réseaux de neurones.
Plusieurs classements des systèmes neuro-symboliques ont été proposés dans la littérature. Ces systèmes sont soit classés suivant leur architecture ou bien selon le degré de couplage des modules et/ou leur type d'intégration (McGarry *et al.*, 1999) et (Wermter et Sun, 2000). Le degré de couplage représente le degré de communication et le flot d'informations entre les modules. Le type d'intégration représente la manière par laquelle les modules sont combinés.
Concernant le degré de couplage, les auteurs de (McGarry *et al.*, 1999) distinguent trois cas de figures (voir la figure 6.1) :

- *Couplage faible (passif) :* les modules communiquent entre eux par l'intermédiaire de fichiers. Une fois que l'un des modules a fini le traitement, il met les résultats obtenus dans un fichier qui sera lu par l'autre module. Le flot de communication est unidirectionnel.
- *Couplage étroit (actif) :* les modules communiquent entre eux par l'intermédiaire d'une mémoire vive et/ou de structures de données partagées et donc, plus d'efforts sont requis pour assurer la synchronisation des modules. Le flot de communication peut être bidirectionnel.

- *Intégration totale (cohabitation) :* la communication entre les modules s'effectue par appels de fonctions. Les modules sont très actifs avec un grand degré d'interaction. Dans une telle intégration, les modules symboliques et neuronaux ne peuvent être différenciés de l'extérieur. Un protocole de communication sophistiqué doit être défini pour contrôler les différents niveaux de communication.

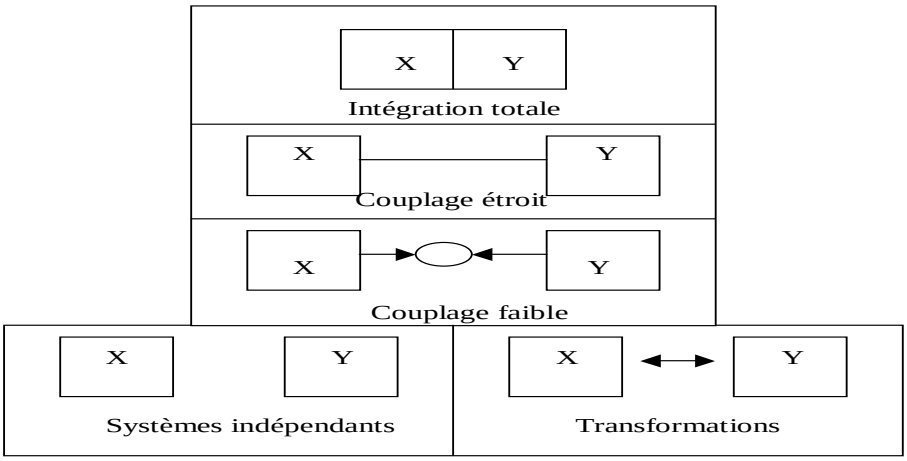


FIG. *6.1: Classification des systèmes hybrides selon leur degré de couplage.*

Un classement selon le mode d'intégration entre les différents modules a été proposé dans (McGarry *et al.*, 1999), (Hilario *et al.*, 1994) et (Hilario *et al.*, 1995), ces modes sont les suivants (voir la figure 6.2) :

- *Traitement chaîné :* dans ce traitement, l'un des modules (symbolique ou connexionniste) est le processeur principal tandis que l'autre effectue les tâches de pré-traitement et/ou de post-traitement. Le traitement est fait de manière séquentielle par un module, puis par l'autre.
- *Sous-traitement :* le traitement principal est assuré par l'un des deux modules qui utilise l'autre pour résoudre des sous-problèmes particuliers.
- *Méta-traitement :* l'un des modules résout le problème, pendant que l'autre effectue le contrôle : c'est le méta-traiteur (contrôleur ou superviseur).
- *Co-traitement :* les deux modules sont au même niveau et coopèrent pour résoudre le problème : chacun des modules peut interagir directement avec l'environnement et peut transmettre et recevoir des informations de l'autre module.

Un autre classement qui intègre en même temps les notions de mode d'intégration et de degré de couplage a été proposé dans (Hilario, 1997) et (Hilario *et al.*, 1995).

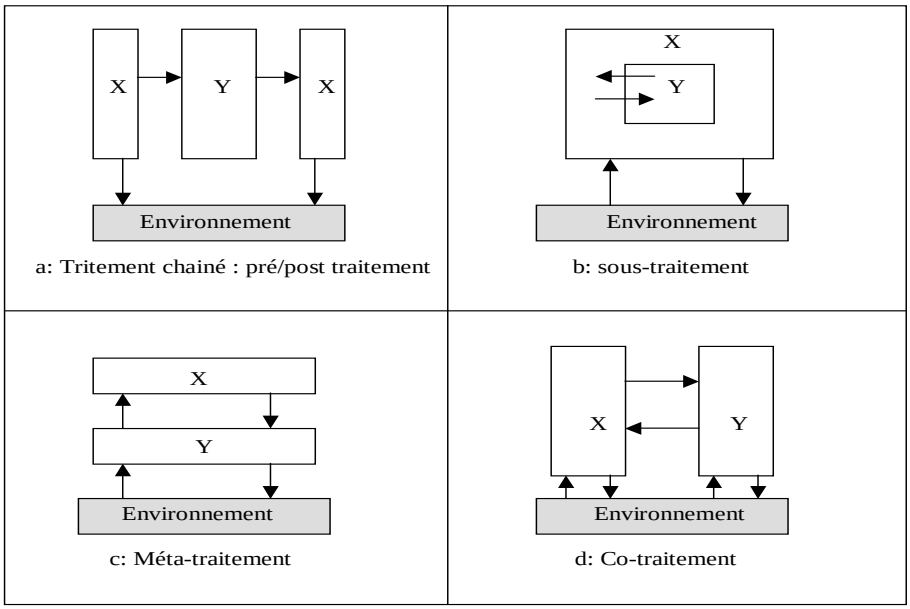


FIG. *6.2: Classification des systèmes hybrides selon les différents modes d'intégration.*

6.2 Systèmes hybrides neuro-RàPC « réseaux de neurones - raisonnement à partir de cas »

Un système neuro-RàPC est un cas particulier des systèmes neuro-symboliques où le module symbolique est le RàPC.
Les réseaux de neurones et le raisonnement à partir de cas ont deux propriétés importantes en commun :

1. Ils sont bien adaptés pour effectuer des tâches de classement et de prédiction. Par exemple, dans le cas des données numériques, un RN produit, généralement, une valeur réelle par neurone de sortie. Dans le problème de classement, il est possible de mettre quelques valeurs de seuils pour interpréter la valeur comme une classe prédite. Un système de RàPC remémore un ensemble de cas similaires au cas cible (l'entrée) et la classe la plus fréquente est sélectionnée (éventuellement, d'autres critères peuvent être utilisés). Pour les tâches de prédiction, la sortie d'un RN peut être directement interprétée comme étant la valeur prédite. Le système de RàPC adapte les solutions des cas remémorés pour fournir la valeur prédite.
2. Les deux méthodes peuvent apprendre de manière continue et donc peuvent être utilisées pour des applications à long terme.

Comme pour les systèmes neuro-symboliques, les systèmes hybrides neuro-RàPC peuvent être classés selon leur degré de couplage (faible, étroit et intégration totale) et selon leur mode d'intégration (traitement chaîné, sous-traitement, méta-traitement et co-traitement).

Les systèmes neuro-RàPC peuvent aussi être classés selon le partage des tâches entre le RN et Le RàPC. Quatre alternatives ont été citées dans (R.B. Sovat, 2001) :

- **Contrôle central** Le RàPC et le RN sont contrôlés par un composant central ;
- **Contrôle distribué** Le contrôle est divisé entre les deux techniques ;
- **Réseau de neurones est dominant** Le contrôle qui domine est celui du RN ;
- **Raisonnement à partir de cas dominant** Le contrôle qui domine est celui du RàPC.

La plupart des systèmes hybrides neuro-RàPC existant utilisent le mode de sous-traitement. Parmi ces systèmes, nous pouvons distinguer ceux où le traitement principal est effectué par le réseau de neurones de ceux où le traitement principal est effectué par le RàPC. Dans le premier cas, le RN est dominant tandis que dans le second cas c'est le RàPC qui est dominant. De plus, dans le dernier cas, les réseaux de neurones peuvent être utilisés dans une ou plusieurs phases du cycle de RàPC. Le réseau de neurones est souvent utilisé dans la phase de recherche pour l'indexation de cas afin d'accélérer la recherche de cas sources (Malek, 2000). Il a aussi été utilisé dans la phase de réutilisation (Corchado et Lees, 2000). Il existe des systèmes où le RN est utilisé dans une phase du cycle de RàPC alors qu'il est dominant dans le système global (Malek, 2000). Dans les systèmes hybrides qui utilisent le mode de co-traitement, le contrôle peut être assuré par un composant central ou divisé entre les deux techniques.

6.2.1 Le système ProBIS

Le système « ProBIS » (Malek, 1996) et (Malek, 2000) est un système hybride intégrant un module de RàPC et un réseau de neurones incrémental à base de prototype (ARN2).
Le module connexionniste sert comme un système d'indexation pour les zones de la mémoire plate. Le RN est donc utilisé pour accélérer la remémoration et pour créer des prototypes représentatifs pour chaque classe. Par conséquent, le mode d'intégration entre les deux modules (module connexionniste et module RàPC) est celui du sous-traitement où le module principal est celui du RàPC.
D'un autre coté, le couplage entre les deux modules est fort, car les deux composants (connexionniste et RàPC) sont intégrés via une structure commune de mémoire : une partie de la mémoire (le haut niveau) appartient au module connexionniste, et une partie (le bas niveau) appartient au module de RàPC. Les deux parties sont liées par des pointeurs qui lient chaque neurone de la couche cachée du réseau à une zone de la mémoire de bas niveau.
Le modèle ProBIS est divisé en trois parties principales :

- *La partie connexionniste :* cette partie est constituée du réseau ARN2 (voir la figure 6.3), ce réseau permet de construire des prototypes représentatifs pour chaque classe, ces prototypes forment le haut niveau de la mémoire. Ils permettent d'organiser les cas dans les différents

groupes de la mémoire plate du bas niveau et ils serviront comme index pour ces différents groupes.

- *La partie mémoire plate :* Cette partie contient la mémoire plate qui est divisée en plusieurs groupes. Chaque groupe est représenté par un prototype dans le réseau. Cette partie contient aussi des méthodes de remémoration qui seront appliquées à certaines parties de la mémoire plate. Ces méthodes sont basées sur les algorithmes des k-plus proches voisins.
- *La partie interface :* cette partie sert à faire l'interface entre les deux parties précédentes, elle comporte trois sous parties :
  - Des liens entre la partie connexionniste et la mémoire plate.
  - Un module de gestion d'interaction qui supervise les deux parties précédentes pendant le processus de mémorisation et de remémoration.
  - Un module de transfert de connaissance qui permet l'échange des connaissances entre le réseau et la mémoire plate.

L'utilisation du RN est dominante dans ce système. Le système de RàPC sert à fournir les solutions aux cas qui ne peuvent pas être résolus par le réseau de neurones utilisé.

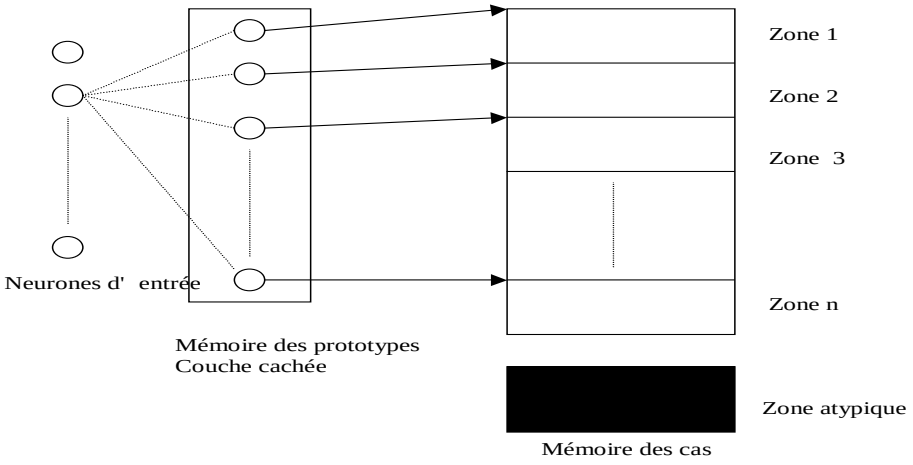


FIG. *6.3: Mémoire du modèle ProBIS.*

6.2.2 Système hybride pour la prédiction de la température de l'océan

Le système de RàPC proposé par (Corchado et Lees, 2000) utilise un réseau de neurones dans la phase de réutilisation du cycle de RàPC (voir la figure 6.4).
Le système a été conçu dans le but d'effectuer des prédictions de la température de l'eau dans l'océan. Un cas dans ce système est décrit par les 40 dernières températures prises dans l'océan et par d'autres informations. La partie solution du cas représente la température de l'eau 5 km plus loin. Ces cas sont construits à partir des données suivant des règles obtenues par les expérimen-

tations. La remémoration du cas le plus proche du cas cible n'est pas adaptée à cette application puisque les données contiennent du bruit. Un ensemble de cas similaires est extrait en utilisant plusieurs mesures de similarité. Ces cas sont utilisés comme entrées d'un réseau de neurones à fonctions de bases radiales (RBF) (Park et Sandberg, 1991). Ce réseau permet de généraliser les solutions des cas remémorés pour fournir la solution du cas cible. Il effectue un apprentissage rapide, a de bonnes capacités de généralisation et apprend sans oubli. Un intervalle d'erreur est associé à chaque valeur prédite.
Comme le montre la figure 6.4, l'acquisition de données est d'abord effectuée en utilisant des capteurs qui permettent de récupérer les températures en temps réel, sur un navire et des images de satellite qui sont reçues chaque semaine. Ces données sont ensuite indexées de façon qu'elles puissent être transformées et gardées dans la base de cas quand c'est nécessaire. Durant la phase de recherche, les k cas les plus similaires au cas cible sont remémorés en utilisant l'algorithme des k plus proches voisins (K-NN). Ces cas sont ensuite utilisés pour effectuer l'apprentissage du RN. La sortie du RN est la température désirée de l'océan obtenue en lui présentant le cas cible (ceci est effectué dans la phase de réutilisation du cycle de RàPC). L'apprentissage du réseau de neurones RBF est réalisé en temps réel pour fournir la prédiction. Durant cette étape, les poids et les centres du RN, utilisés dans les prédictions précédentes, sont remémorés à partir de la base de connaissances et adaptés en utilisant l'ensemble d'apprentissage. Dans la phase de révision, la prédiction finale est obtenue en modifiant la prédiction proposée par le RN. Ceci est effectué en tenant compte de la précision des prédictions précédentes. A chaque cas est associée une erreur moyenne. C'est une mesure de l'erreur moyenne des prédictions précédentes pour lesquelles ce cas a été utilisé pour l'apprentissage du RN. Les seuils des erreurs à ne pas dépasser sont calculés en effectuant la moyenne des erreurs moyennes des cas utilisés pour l'apprentissage du RN. Dans la phase d'apprentissage, deux tâches sont effectuées :

- La structure interne (poids des centres) du RN est sauvegardée après chaque prédiction (après l'apprentissage du RN).
- Certains paramètres qui constituent les cas sont modifiés.

Toutes les prédictions effectuées durant les 5 derniers kilomètres et tous les cas utilisés pour l'apprentissage du RN sont sauvegardés dans la base de données. Ces prédictions sont ensuite comparées aux valeurs réelles correspondantes de la température de l'eau. Les erreurs de prédiction sont alors utilisées pour modifier la pertinence des cas, pour réduire la base de cas et déterminer les seuils d'erreurs. Le mode d'intégration entre les modules (RàPC et RN) est celui du sous-traitement où le RàPC est dominant.

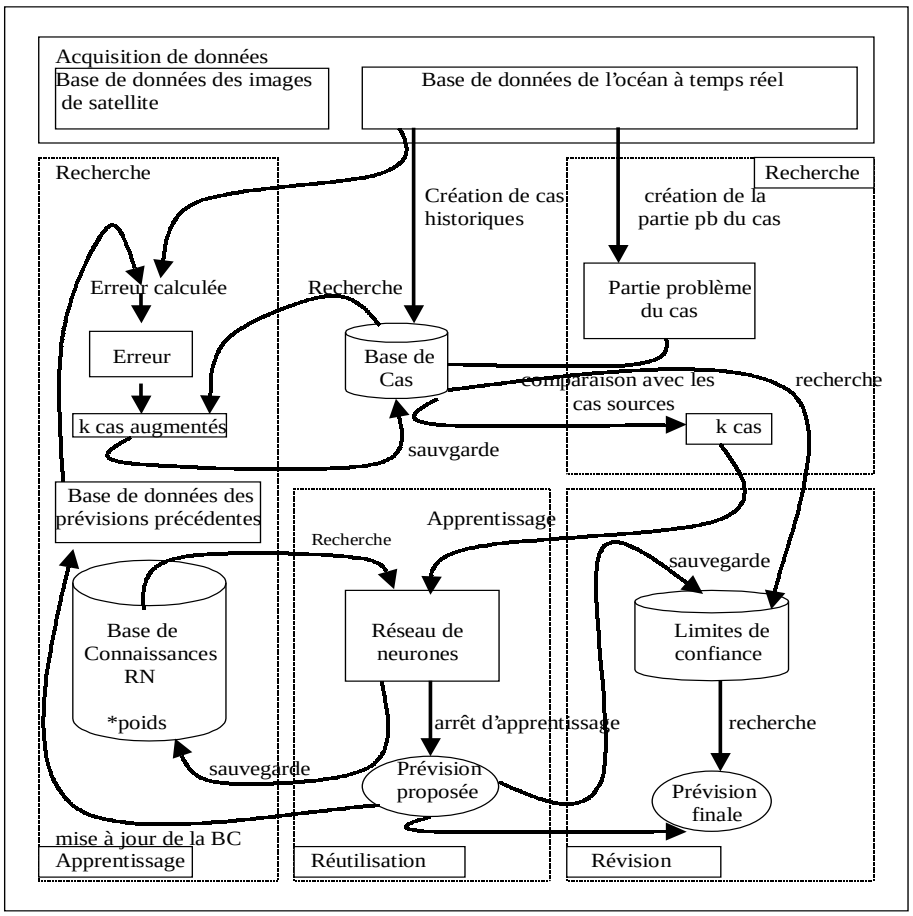


FIG. *6.4: Architecture du système hybride pour la prédiction de la température de l'océan.*

6.2.3 Système hybride pour la prévision de marées rouges

Le système de RàPC proposé dans (Fdez-Riverola et Corchado, 2003) est une extension du système précédent, il a été utilisé pour la prévision des marées rouges dans certaines régions côtières (voir la figure 6.5). Il contient deux réseaux de neurones différents : le réseau GCS (Fritzke, 1994) et le réseau RBF (Park et Sandberg, 1991). Le module RàPC et les deux réseaux de neurones sont intégrés par le mode de sous-traitement où le GCS est utilisé pour l'indexation et le réseau RBF est utilisé dans la phase de réutilisation pour fournir des prévisions. Le module dominant est, bien sûr, celui du RàPC.

La phase de recherche est effectuée en utilisant le réseau de neurones « Growing Cell Structures (GCS) » (Fritzke, 1994). Celui-ci effectue une indexation automatique des cas et accélère la remémoration des cas sources les plus similaires au cas cible. De plus, c'est un réseau évolutif qui tient compte de l'arrivée dynamique des données. L'adaptation de cas est effectuée en utilisant le réseau RBF (Park et Sandberg, 1991) comme dans le système précédent (Corchado et Lees, 2000). La révision est effectuée en utilisant un groupe de systèmes flous autonomes.

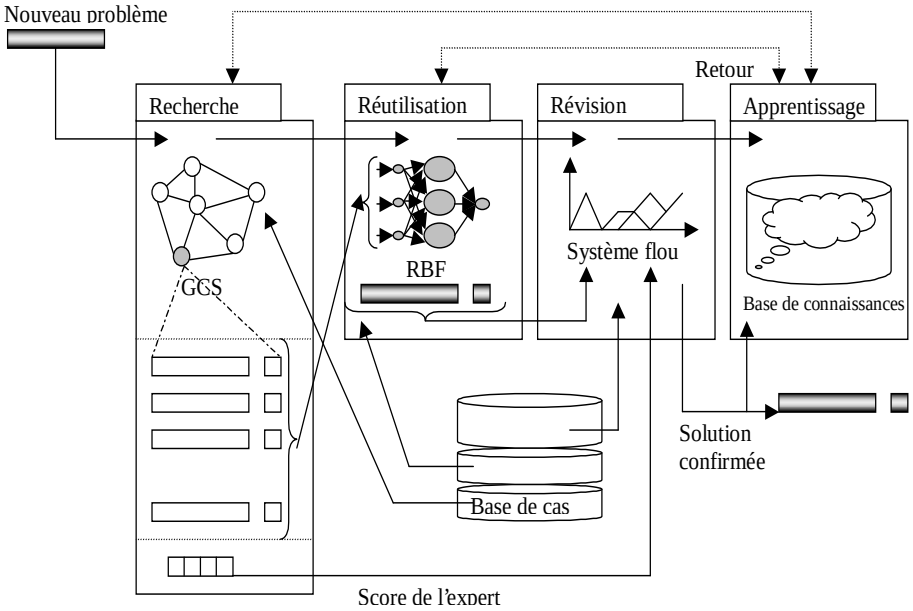


FIG. *6.5: Architecture du système hybride pour la prédiction des marées rouges.*

De même que dans le système précédent, le mode d'intégration est celui du sous-traitement où le RàPC est dominant.

6.2.4 Système hybride pour le diagnostic de pannes

Le système décrit dans (Jha *et al.*, 1999) a été développé dans le but d'effectuer un diagnostic des pannes de machines dans un service d'aide aux consommateurs. La base de données du service des consommateurs garde toutes les traces des problèmes rencontrés ainsi que des solutions qui leurs sont apportées sous forme d'enregistrements. Chaque enregistrement contient des informations sur le consommateur et sur les détails du diagnostic effectué. Deux ensembles d'enregistrements sont construits. Le premier contient tous les services fournis par les ingénieurs durant les années précédentes. Comme les mêmes conditions de pannes et les solutions qui leurs sont apportées peuvent se répéter, un deuxième ensemble est créé. Celui-ci est un sous ensemble du premier dans lequel les cas répétés sont supprimés.

Dans ce système, un réseau de neurones est utilisé pour effectuer la phase de remémoration (voir la figure 6.6). Le mode d'intégration est alors celui du sous-traitement où le RàPC est dominant.
Après le pré-traitement des données de la base du service des consommateurs, le deuxième ensemble est utilisé pour initialiser le réseau de neurones LVQ3 (McDermott, 1990). Le premier ensemble est ensuite utilisé pour l'apprentissage du réseau.
Lors de l'utilisation du système (voir la figure 6.6), l'entrée de l'utilisateur est d'abord pré-traitée.

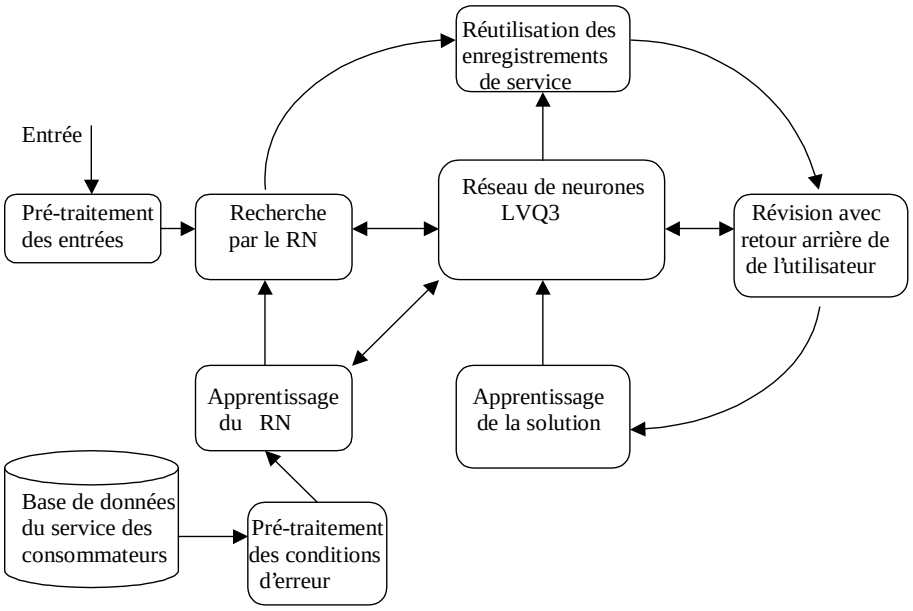


FIG. *6.6: Système de diagnostic d'erreurs*

Un pré-traitement des conditions de défauts du service d'enregistrements de la base de données dans le service du consommateur est effectué afin d'extraire des mots-clés et des phrases-clés. Ceci inclue le pré-traitement des expressions qui représentent une condition de défauts. Ensuite, la recherche de cas similaires est effectuée par le réseau de neurones et un ensemble de cas (panne, solution) triés suivant leurs distances au cas cible est remémoré.
La réutilisation consiste à suivre les consignes données par les solutions associées aux cas remémorés ordonnés qui sont présentés à l'utilisateur dans l'ordre.
Dans la phase de révision, l'utilisateur suit les consignes des solutions données pour chaque cas remémoré dans l'ordre qui lui a été présenté. Si son problème est réglé par l'un des cas suggérés, celui-ci est marqué pour augmenter sa confiance et lui donner la priorité dans les prochains diagnostics. Sinon, des ingénieurs sont consultés pour résoudre le problème.
Dans la phase d'apprentissage, le nouveau cas (panne, solution) est retenu par le système. Celui-ci met aussi à jour les poids du réseau de neurones.

6.2.5 Système hybride pour l'exploration de données

Le système hybride proposé dans (Shin et Park, 2000) est utilisé pour résoudre certain problèmes pratiques de la fouille de données. L'un de ces principaux problèmes est celui de la modélisation des comportements dynamiques qui changent avec le temps et de la prédiction à partir de cette modélisation. Ce système contient deux modules : connexionniste et RàPC (voir figure 6.7)

qui coopèrent pour fournir une prédiction. Cette prédiction est effectuée par le « gestionnaire de requêtes » (*query manager* (PQM)). Le PQM reçoit de nouvelles requêtes, il consulte le RN et le système de RàPC de manière parallèle. Quand les deux prédicteurs donnent la même valeur, le PQM retourne la valeur prédite. Quand les résultats fournis sont considérablement différents, le PQM indique que la décision ne peut être prise par le système et que l'intervention d'experts est requise pour les cas rejetés.

Après la phase d'apprentissage du réseau de neurones, celui-ci peut fournir les poids des caractéristiques des cas qui sont utilisés dans le module de RàPC. La base de cas permet aussi de fournir un ensemble d'apprentissage pour le réseau de neurones.

Donc, le mode d'intégration dans ce système est celui du co-traitement. Les deux modules (RN et RàPC) coopèrent de manière égale pour fournir la solution du cas cible. Le contrôle dans ce système est central, il est effectué par le PQM.

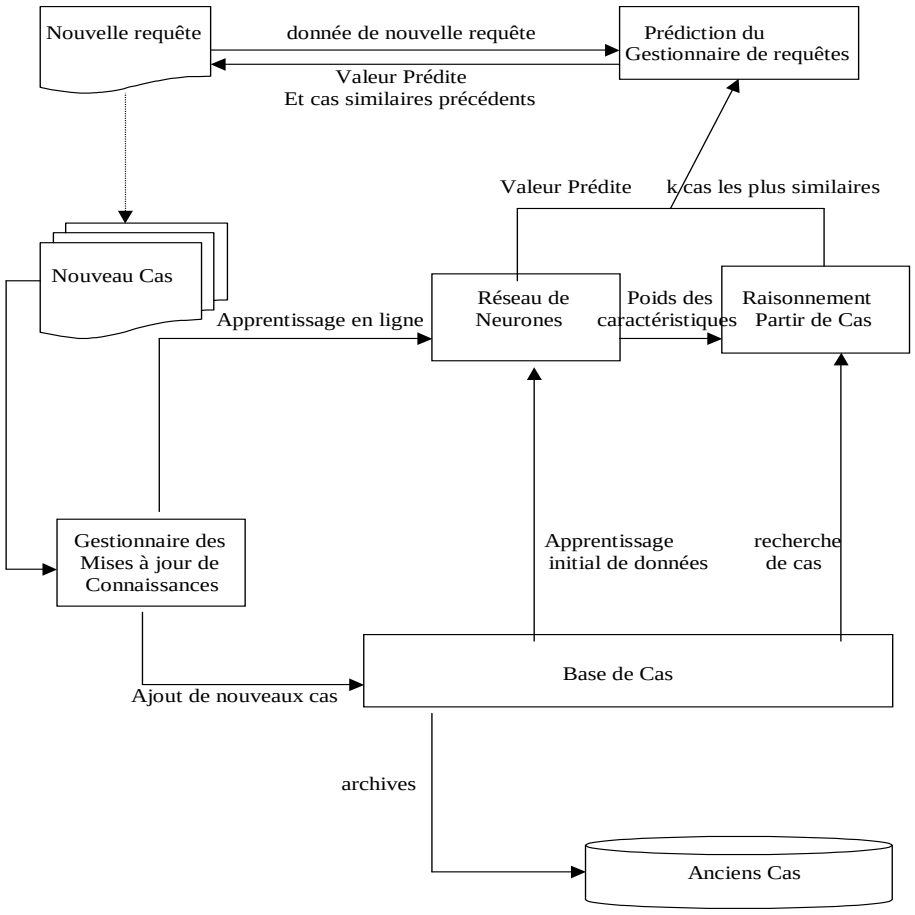


FIG. *6.7: Architecture du système hybride pour l'exploration de données.*

6.2.6 Système hybride pour la supervision des ventes

Le système hybride proposé dans (Murray-Smith et Thakar, 1993) est utilisé dans le domaine commercial pour la supervision des ventes. Il combine un réseau de neurones de type RBF (Park et Sandberg, 1991) avec le raisonnement à partir de cas (voir la figure 6.8). Le réseau RBF est utilisé dans la phase de réutilisation pour fournir des solutions aux cas cibles.
La particularité de ce système est qu'il transforme les caractéristiques symboliques en caractéristiques numériques en utilisant la logique floue ou en utilisant des connaissances *a priori* pour définir des similarités. Donc un pré-traitement des entrées est effectué avant l'utilisation du réseau RBF qui fournit une solution qui est ensuite transformée (l'inverse du traitement d'entrée est effectué).

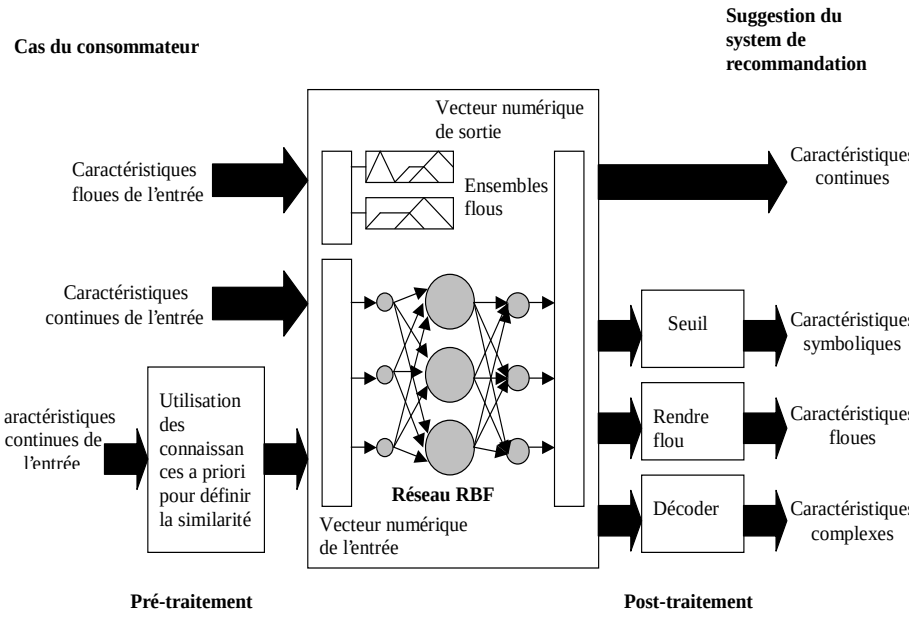


FIG. *6.8: Diagramme de la mémoire du système de supervision de ventes et du réseau de neurones.*

6.2.7 Système hybride pour la modélisation de comportements

Les auteurs de (Mujica et Vehi, 2003) ont proposé un système de RàPC où la base de cas est représentée par une carte auto-organisatrice (SOM) (voir la figure 6.9).
Ce système est utilisé pour modéliser les comportements des clients d'une banque. Un cas dans ce système est représenté par le vecteur poids d'un neurone de la carte SOM. Ceux-ci sont obtenus après l'apprentissage de SOM. Lorsqu'un nouveau cas se présente au système, un groupe de cas sources similaires au cas cible est remémoré à partir des neurones de SOM. Comme les cas remémorés peuvent appartenir à différentes classes, les auteurs proposent trois méthodes pour

adapter et réviser la solution. La classe du cas cible est déterminée par :

- La classe dominante : c'est la classe la plus fréquente dans le groupe.
- La valeur minimale de la mesure HEOM : c'est la classe du cas source ayant la valeur minimale de la HEOM (Heterogeneous Euclidean-Overlap Metric). Cette mesure utilise une distance hétérogène qui utilise différentes distances sur différents types d'attributs (continus et nominaux : qualitatifs ou quantitatifs). La fonction HEOM entre deux valeurs x et y est donnée par :

$$HEOM(x,y) = \sqrt{\sum_{a=1}^{m} d_h(x_a, y_a)^2} \tag{6.1}$$

où

$$d_h(x_a, y_a) = \begin{cases} 1, & si\ x_a\ ou\ y_a\ est\ inconnu. \\ overlap(x_a, y_a), & si\ a\ est\ symbolique. \\ \frac{|x_a - y_a|}{max_a - min_a}, & sinon \end{cases}$$

et

$$overlap(x_a, y_a) = \begin{cases} 0, & si\ x_a = y_a \\ 1, & si\ x_a \neq y_a \end{cases}$$

- Hybride : la classe dominante si le domaine représente plus de 75 %. La valeur minimale de HEOM sinon.

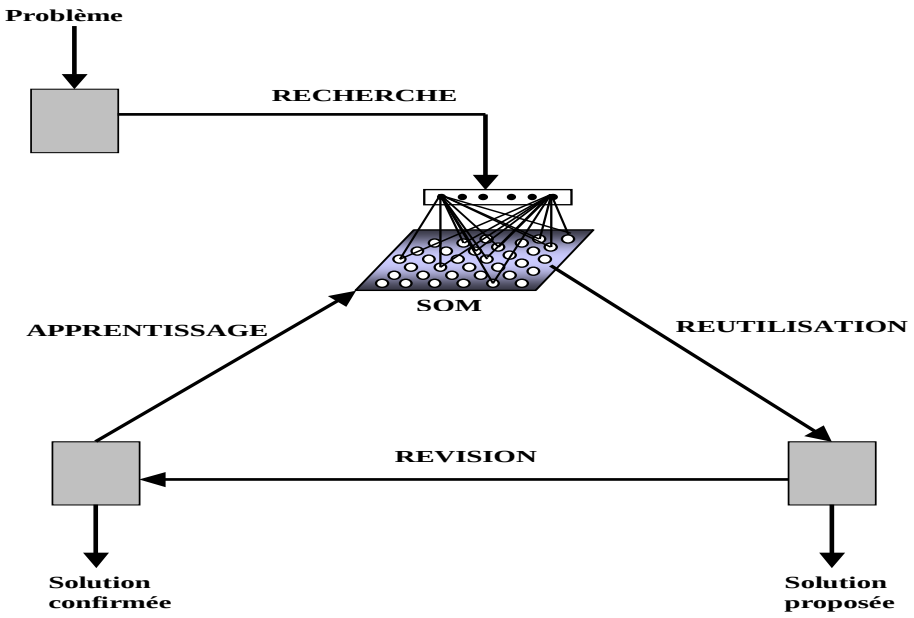


FIG. *6.9: Cycle RàPC du système hybride pour la modélisation de comportements.*

Dans ce système, la base de cas représente la mémoire du RN. Le mode d'intégration est celui du sous-traitement où le RàPC domine.

6.2.8 Système hybride utilisant une mémoire associative

Dans (R.B. Sovat, 2001), le RN est utilisé dans les phases de recherche et d'adaptation de cas. Dans le système proposé, un réseau de neurones MLP (Rosenblatt, 1958) dont l'apprentissage est effectué par l'algorithme de *Backpropagation* est utilisé. L'utilisation du RN possède quelques inconvénients malgré les différents avantages qu'il fournit.

- Le premier est le besoin de refaire l'apprentissage de manière périodique. Ceci se produit quand le RN effectue un apprentissage en utilisant une base de cas initiale. Quand le système commence à apprendre de nouveaux cas, le mécanisme tend à être imprécis.
- Le second problème vient de l'incapacité du RN utilisé à ordonner les solutions. Le RN peut choisir un cas à partir de la base de cas mais ne peut pas donner la deuxième ou la troisième meilleure suggestion.

Afin de trouver une solution pour ces deux problèmes, deux techniques sont utilisées en parallèle : un ensemble de réseaux de neurones (dans le modèle, un seul RN est utilisé plusieurs fois) et l'algorithme du plus proche voisin (K-NN). Le K-NN est utilisé pour deux buts : ajuster la sortie du RN et produire un ordre optionnel si le premier cas sélectionné n'est pas approuvé par l'utilisateur. Le fonctionnement de la mémoire est montré dans la figure 6.10.

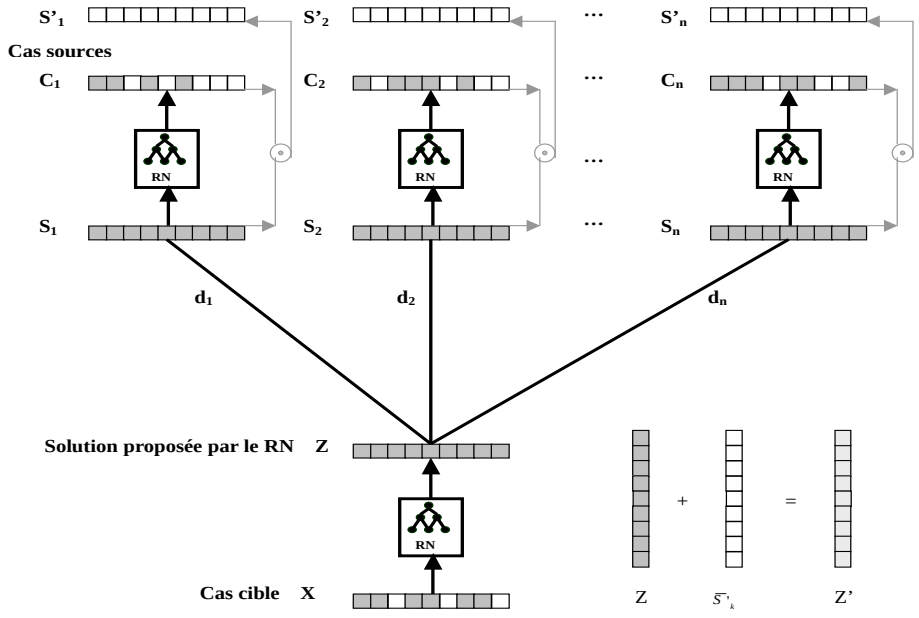


FIG. *6.10: Fonctionnement de la mémoire associative du système hybride.*

Suivant les suggestions proposées par la mémoire utilisée, le vecteur de sortie z du RN obtenu à partir du cas cible x peut ne pas être la valeur finale et nécessiter des ajustements. Alors un groupe de n cas (sélectionnés en utilisant l'algorithme K-NN) à partir de z est aussi utilisé comme entrée du RN, générant ainsi un ensemble de cas $s_i \ \ (i = 1, ..., n)$. Comme ces cas ont été utilisés pour l'apprentissage du RN, les vecteurs de sortie $c_i \ \ (i = 1, ..., n)$ devraient être les mêmes que les solutions stockées dans la base de cas. Si ce n'est pas le cas, les anomalies sont stockées dans les vecteurs $s'_i \ \ (i = 1, ..., n)$.
A ce stade du processus, l'algorithme K-NN est capable d'obtenir l'ordre des cas juste en calculant les distances $d_i \ \ (i = 1, ..., n)$ entre les vecteurs $s_i \ \ (i = 1, ..., n)$ et le vecteur z. Cependant, afin d'obtenir le meilleur ajustement de la solution initiale proposée, la solution finale est le vecteur z'. Ce vecteur est la somme du vecteur z et du vecteur $\bar{s}'_k$, qui est une moyenne arithmétique des k plus proches vecteurs $s'_i \ \ (i = 1, ..., k)$ fournis par K-NN.
La correction produite par K-NN crée un second niveau de connaissances qui complète les connaissances générales stockées dans le RN. Le RN effectue la première sélection. Plus la base de cas courante est proche de la base de cas utilisée pour l'apprentissage du RN, plus la cette sélection est bonne. Cette méthode compense le problème de non mise à jour du RN en temps réel. L'ordre des cas construit par K-NN peut être utilisé pour resélectionner des cas si c'est nécessaire. Chaque cas peut être présenté à l'entrée du RN dans l'ordre précédemment fourni.
Dans ce système, le mode d'intégration du RN est celui du sous-traitement puisqu'il est utilisé dans la phase de recherche et celle de réutilisation et le module dominant est toujours le RàPC.

6.2.9 Système hybride de recommandation par filtrage collaboratif

Dans le système hybride proposé dans (Roh *et al.*, 2003), une carte auto-organisatrice SOM effectue une classification des cas de la base de cas et fournit pour chaque groupe formé un cas représentant. Ce cas est le vecteur poids du neurone obtenu après plusieurs adaptations lors de la phase d'apprentissage de la carte SOM.

Le processus de classification-indexation est composé de trois étapes (voir la figure 6.11) : le *profiling*, l'inférence et la prédiction.
Dans l'étape de *profiling*, une analyse par composantes principales est effectuée pour réduire la dimension de l'entrée et pour déterminer le nombre de neurones de la carte SOM. Les poids des neurones de la carte SOM sont ensuite initialisés et un apprentissage de la carte est effectué. Tous les utilisateurs représentés dans la phase d'apprentissage de SOM sont liés aux cas représentants dont les valeurs des caractéristiques sont données par les poids des neurones de SOM.
Dans l'étape d'inférence, la RàPC compare l'utilisateur courant aux cas représentants. Le groupe des cas sources les plus similaires au cas cible est obtenu à partir du liens du cas représentant aux cas de la base de cas.
L'étape de prédiction consiste à fournir la prédiction demandée en utilisant la mesure de cor-

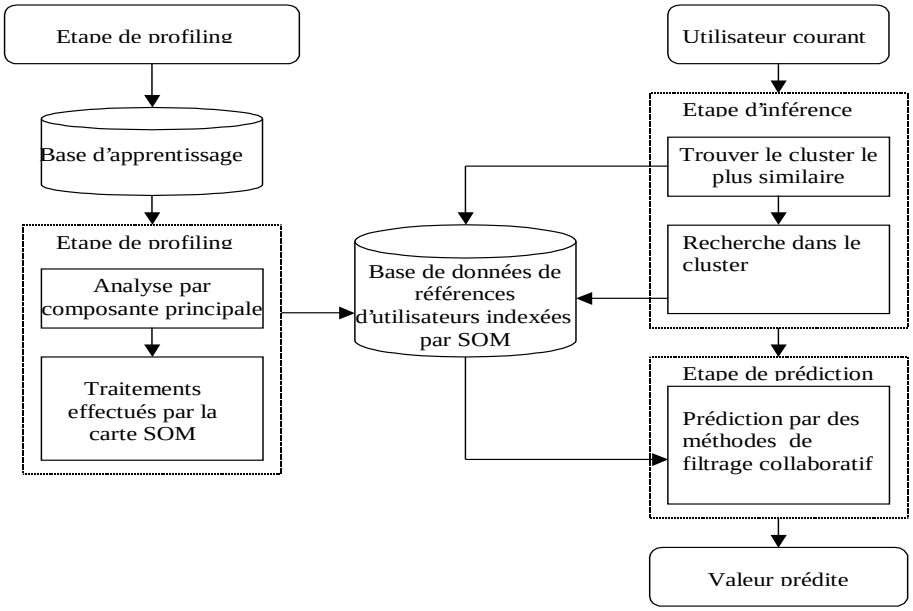


FIG. *6.11: Architecture du système de de recommandation par filtrage collabortif.*

rélation de Peterson (Shardanand et Maes, 1995) qui est une distance utilisée dans le filtrage collaboratif (Roh *et al.*, 2003) entre l'utilisateur courant et les utilisateurs se trouvant dans le groupe sélectionné.
Comme pour la plupart des systèmes cités, le mode d'intégration est celui du sous-traitement où le RàPC est dominant.

6.2.10 Discussion

Parmi les systèmes hybrides que nous avons décrit dans le paragraphe §6.1, seuls les systèmes proposés dans (Corchado et Lees, 2000) et (Fdez-Riverola et Corchado, 2003) traitent les séquences temporelles. La représentation des cas dans ces systèmes est une représentation à base d'instants où un cas est une expérience précise dans une séquence de longueur fixée. Les réseaux de neurones utilisés sont le réseau RBF dans la phase d'adaptation de (Corchado et Lees, 2000; Fdez-Riverola et Corchado, 2003) et le réseau GCS dans la phase de recherche pour l'indexation de cas (Fdez-Riverola et Corchado, 2003). Ces réseaux ne sont pas conçus pour traiter les séquences temporelles et donc considèrent les cas comme des vecteurs d'attributs-valeurs. Ceci constitue une limite de ces deux systèmes dans plusieurs applications.
Certains systèmes qui utilisent un RN pour l'indexation des cas de la base de cas prennent en compte l'arrivée dynamique des données (Malek, 2000), (Park et Sandberg, 1991) et (Fdez-Riverola et Corchado, 2003). Ceci a été réalisé en choisissant des RN incrémentaux pour indexer les cas. Ce choix permet d'éviter de refaire à chaque fois l'apprentissage du réseau de neurones en utilisant la totalité des données. Mais les réseaux utilisés, malgré le fait qu'ils s'adaptent aux nouvelles données, n'assurent pas la préservation des anciennes déjà apprises. Les connaissances

acquises par ces réseaux durant les phases d'apprentissage précédentes peuvent être perdues. Quand le RN est utilisé pour indexer les cas, sa mise à jour se fait de façon périodique. L'arrivée de nouveaux cas n'est pas prise en compte par le réseau en temps réel contrairement au fonctionnement habituel d'un système de RàPC. Seuls les systèmes (Malek, 2000) et (R.B. Sovat, 2001) ont proposé des solutions pour pallier à ce problème. Dans (Malek, 2000), les nouveaux cas ajoutés au système sont mis dans la partie atypique de la base de cas et sont rendus, tout de suite après leur ajout, disponibles pour résoudre les nouveaux cas cibles. Dans (R.B. Sovat, 2001), une méthode qui estime l'erreur commise dans la prédiction lorsque le réseau de neurones n'est pas encore mis à jour est proposée.

Nous présentons dans le chapitre suivant (chapitre 7), un système hybride qui effectue le classement des séquences et qui utilise deux réseaux de neurones M-SOM-ART, un pour indexer les cas et l'autre dans la phase de réutilisation pour fournir les solutions du cas cible. L'organisation de la mémoire a été inspirée du modèle de (Malek, 2000) pour obtenir un système incrémental en temps réel.

Chapitre 7

CASEP2 : Système hybride pour le traitement des séquences

Nous présentons dans ce chapitre le système hybride appelé « CASEP2 », qui combine le raisonnement à partir de cas (RàPC) avec des réseaux de neurones (RN), pour le classement (ou la prédiction) de séquences (Zehraoui *et al.*, 2004). Dans CASEP2, un cas est modélisé par une matrice de covariance dynamique qui prend en compte la distribution des états de la séquence dans l'espace ainsi que leur ordre temporel. La base de cas du système est partitionnée en utilisant un réseau de neurones adéquat.

7.1 Architecture du système CASEP2

Dans le système CASEP2 (voir la figure 7.1), un réseau de neurones M-SOM-ART permet d'indexer les cas de la base de cas et un autre permet d'effectuer des tâches de classement (ou de prédiction).
Comme dans le système ProBIS (Malek, 1996) et (Malek, 2000), la base de cas est divisée en plusieurs parties. Chaque partie est indexée par un neurone sauf une seule appelée partie atypique. Celle-ci va contenir les cas ajoutés à la base de cas durant l'utilisation du système.

Le système CASEP2 fonctionne selon deux modes :

- *Le mode de construction hors ligne :*
  Un apprentissage est effectué par le réseau de neurones. Celui-ci permet de construire les prototypes et d'indexer la base de cas en la partitionnant et en reliant chaque partie (sauf la partie atypique) à un neurone. Une réduction de la base de cas est aussi effectuée par le système.
- *Le mode d'utilisation en ligne :*
  Lorsqu'un nouveau cas se présente, un neurone est activé dans le RN. Si ce neurone est lié

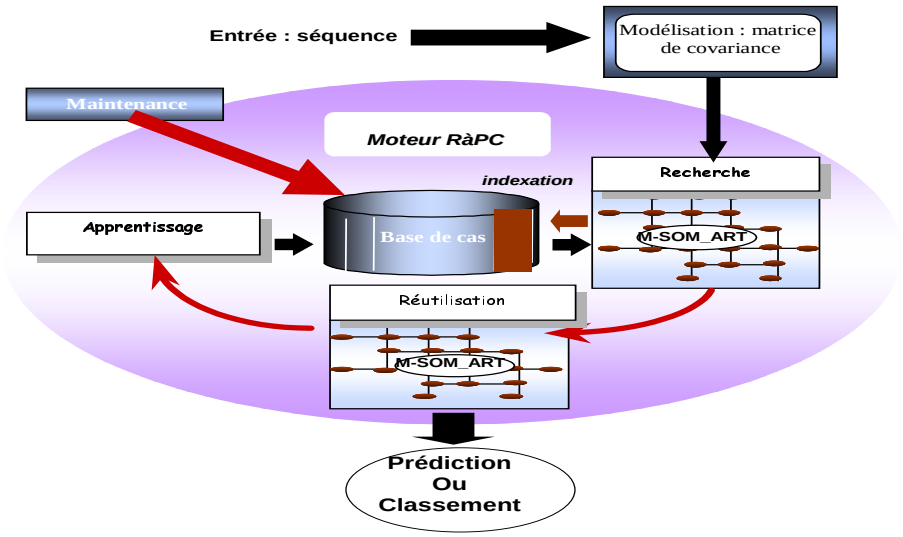


FIG. *7.1: Architecture du système CASEP2.*

à une partie de la base de cas, alors une recherche de cas similaires au cas cible est effectuée dans cette partie. Sinon la recherche est effectuée dans la partie atypique. De nouveaux cas sont ajoutés à la base de cas durant l'utilisation du système. Ceux-ci seront utilisés pour former une base d'apprentissage du RN durant la phase de construction.

Les différentes phases du cycle de RàPC sont décrites dans ce qui suit.

7.2 Description du réseau connexionniste utilisé

Le réseau de neurones utilisé est le réseau M-SOM-ART (voir le paragraphe §5.3) qui possède les propriétés suivantes :

- effectue des classements (ou des prédictions) et des classifications ;
- traite des données temporelles ;
- possède les propriétés de plasticité et de stabilité.

Ce réseau est une version temporelle évolutive de la carte SOM (Kohonen, 1995) qui utilise le paradigme de la théorie de résonance adaptative (*Adaptive Resonance Theory* : ART) (Carpenter et Grossberg, 1988). Celui-ci permet de doter le réseau M-SOM-ART des propriétés de stabilité et de plasticité (voir le paragraphe §5.1). La plasticité concerne l'acquisition en continu des nouvelles connaissances[14] et la stabilité concerne la préservation des connaissances déjà acquises. Le modèle ART permet de contrôler l'évolution du réseau de neurones en introduisant un test de vigilance qui vérifie si le neurone activé est assez proche de l'entrée. L'aspect temporel des données est

[14]Les connaissances dans ce contexte sont le résultat de la phase d'apprentissage du réseau de neurones.

pris en compte en modélisant toute séquence par une matrice de covariance dynamiques (voir le paragraphe §5.2).
M-SOM-ART effectue une classification de séquences qui consiste à former des groupes de cas similaires pour l'indexation de la base de cas et des classements (ou des prédictions) de séquences pour fournir les solutions des cas cibles. L'aspect temporel des données est pris en compte et les propriétés de stabilité-plasticité sont importantes pour une utilisation à long terme du système.

7.3 Description générale

Nous allons d'abord commencer par introduire quelques définitions utiles.
Une séquence q est une succession ordonnée d'états. Elle décrit une succession d'événements instantanés (dans notre application, une séquence représente la navigation d'un utilisateur sur le site Web).
Un état $E_{jq} = (v_i),\ 1 \leq i \leq n$, représente une étape unitaire dans le temps de la séquence q (dans notre application, un état représente des informations sur une page visualisée par l'internaute). Il est caractérisé par un ensemble de valeurs $v_i (1 \leq i \leq n)$ prises par n variables $c_i (1 \leq i \leq n)$ appelées caractéristiques et par sa position j dans la séquence q.
Notre objectif est de fournir la valeur d'une propriété S d'un ensemble d'états succédant l'état courant d'une séquence en utilisant des séquences qui décrivent des successions d'événements similaires. Cette propriété peut représenter, par exemple, une caractéristique de l'état suivant l'état courant ou bien un classement des séquences (dans notre application, les classes représentent le profil de l'utilisateur du site : {acheteur, non acheteur}). Pour ceci, nous disposons d'un grand nombre de séquences de longueurs variables.

Structure du cas cible

Dans « CASEP2 », une séquence $q(m) = (E_{jq}),\ 1 \leq j \leq m$, est modélisée par une matrice de covariance dynamique cov_q (voir le paragraphe §5.1) donnée par :

$$cov_q(m) = \frac{1}{p}[E_{1q}E_{1q}^T + \sum_{j=2}^{m}(E_{jq} - \bar{q}(j))(E_{jq} - \bar{q}(j))^T]$$

où $\bar{q}(j)$ représente la moyenne dynamique des états de la séquence et X^T représente la transposée du vecteur X.
La partie problème du cas est définie par la matrice de covariance dynamique associée à la séquence et la partie solution est la classe de la séquence (ou la valeur prédite).
La partie problème du cas cible représente la séquence courante[15], qui est modélisée par une matrice de covariance dynamique.
Ce modèle est fourni par le réseau de neurones M-SOM-ART. Il permet de représenter chaque

[15]Une séquence est formée à chaque présentation d'un nouvel état. La séquence courante est une sous-séquence (une partie de la navigation d'un internaute dans notre application) de la séquence finale entière

séquence et ses sous-séquences par des matrices de même dimension. Ceci permet d'éviter l'extraction de sous-séquences. De plus, dans la séquence courante $q(m+1)$ correspondant au cas cible, la nouvelle matrice de covariance $cov_{q(m+1)}$ peut être calculée en utilisant la précédente $cov_{q(m)}$ associée à la même séquence $q(m)$ à chaque présentation d'un nouvel état comme suit :

$$cov_{q(m+1)} = \frac{m}{m+1} cov_{q(m)} + \frac{1}{m+1}[(E_{(m+1)q} - \bar{q}(m+1))(E_{(m+1)q} - \bar{q}(m+1))^T]$$

Organisation de la mémoire

La mémoire de CASEP2 est constituée de deux niveaux (voir la figure 7.2) :

- La mémoire qui contient les cas prototypiques (prototypes) : elle est utilisée durant la phase de recherche comme un système d'indexation afin de réduire le temps de recherche dans la base de cas. Chaque prototype est représenté par un neurone, qui peut indexer un ensemble de cas (une partie de la base de cas). Les neurones ne sont pas tous liés à la base de cas.
- La mémoire qui contient des cas réels (la base de cas) : c'est une simple mémoire plate dans laquelle les cas sont organisés en parties contenant des cas similaires. Elle est partitionnée par le réseau de neurones. Chaque partie, sauf une seule, est liée à un neurone. La partie, qui n'est pas liée au réseau de neurones (la partie atypique) contient les cas ajoutés à la base de cas durant l'utilisation du système.

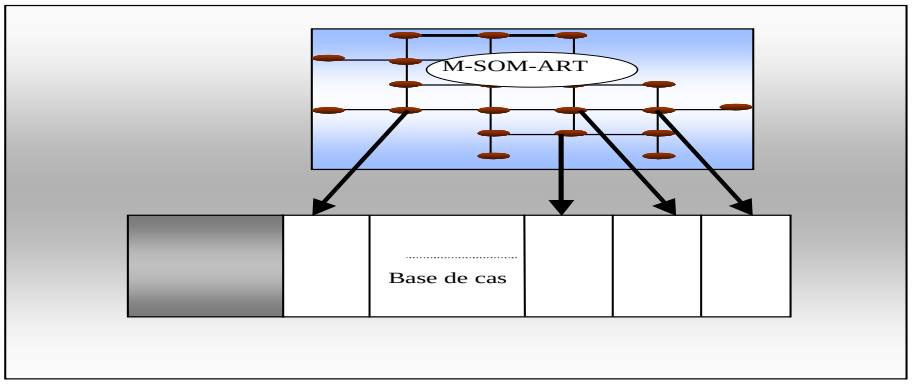


FIG. *7.2: Organisation de la mémoire.*

Le fait d'utiliser un réseau de neurones permet d'améliorer l'efficacité de remémoration. L'utilisation de la base de cas réels pourrait permettre d'avoir une réponse plus précise que celle obtenue par le réseau de neurones qui ne garde en mémoire que les représentants des cas qui ont

activé les neurones et de justifier les résultats fournis par le systèmes.

7.4 Le mode de construction

Dans cette phase, l'apprentissage du réseau de neurones, la construction de la base de cas et sa réduction sont effectués.

Initialisation du système CASEP2
Initialement, la base de cas est vide. Une base d'apprentissage est utilisée pour la remplir en créant des parties liées à des neurones. Cette base d'apprentissage contient un ensemble de séquences codées en matrices de covariances dynamiques.
Lors de la dernière étape de la phase d'apprentissage du réseau de neurones, la base de cas est construite. A chaque présentation de la séquence courante, un neurone est activé parmi les prototypes du RN. A chaque neurone activé est associée une partie de la base de cas. Le cas qui a activé ce neurone est ajouté à la partie qui lui est liée. Des mesures de maintenance (voir le chapitre 3) sont initialisées et associées aux cas ajoutés à la base de cas.

Mise à jour du système
Après l'initialisation du système, celui-ci est utilisé pour fournir les solutions des cas cibles (mode d'utilisation). Durant cette phase d'utilisation, des cas sont ajoutés à la partie atypique de la base de cas. Ceci permet leur utilisation juste après la phase d'apprentissage du cycle de RàPC. Quand la taille de cette partie dépasse un seuil γ, le système est mis à jour dans le mode de construction durant lequel un apprentissage du RN est effectué en utilisant les cas de cette partie de la base de cas. Certains cas seront ajoutés à certaines parties (un cas est ajouté à la partie liée au neurone le plus similaire au cas dépassant le seuil de vigilance). De nouvelles parties peuvent être créées pour contenir d'autres cas.

Phase de réduction
Dans cette phase, la réduction de la base de cas est effectuée de la même manière que dans le système CASEP (voir le chapitre 3). Les cas non utiles sont supprimés, ceci permet de contrôler le contenu de la base de cas. La suppression des cas obsolètes pourra aussi être envisagée pour une utilisation à long terme du système.

7.5 Le mode d'utilisation

A chaque présentation d'un état de la séquence courante, celle-ci est modélisée par une matrice de covariance. Un neurone est ensuite activé dans le réseau de neurones.
Si la confiance de la solution associée au neurone activé est plus grande qu'un paramètre β, la solution est retournée par le système.

Sinon, le module RàPC est utilisé. Nous décrivons ci-dessous le cycle de RàPC dans CASEP2.

Phase de recherche

- Si le neurone activé est lié à une partie de la base de cas, une recherche de cas similaires au cas cible dans cette partie est effectuée.
- Sinon la recherche est effectuée dans la partie atypique.

La similarité entres deux cas modélisés par les matrices de covariances cov_x et cov_y est inversement proportionnelle à la distance de Frobenius (voir la paragraphe §5.2), elle est définie comme suit :

$$similarite(cov_x, cov_y) = \frac{1}{1 + df(cov_x, cov_y)}$$

Les α cas les plus similaires au cas cible dont la similarité dépasse un certain seuil sont remémorés. Le seuil de similarité est une fonction de l'étendue de chaque cas (voir le chapitre 3).

Phase de réutilisation

Dans cette phase, un autre réseau de neurones M-SOM-ART est utilisé pour fournir les prédictions (ou les classements). Celui-ci prend les α cas remémorés dans la phase de recherche comme base d'apprentissage. Ensuite, une solution est fournie pour le cas cible. Afin d'obtenir un réseau de neurones de taille raisonnable (ceci est nécessaire pour satisfaire les contraintes temps réel), M-SOM-ART est initialisé pour chaque présentation d'une nouvelle séquence (correspondant à une nouvelle navigation d'un internaute dans notre application). Lors du traitement d'une séquence, le même réseau de neurones est utilisé jusqu'à ce qu'elle se termine. Ceci permet la réutilisation des connaissances déjà acquises à partir des traitements effectués dans la même séquence.

Phase d'apprentissage

Dans cette phase, seuls les cas cibles pour lesquels la solution fournie n'est pas correcte[16] sont ajoutés à la partie atypique de la base de cas. Ceci permet d'augmenter le recouvrement de la base de cas pour que le système puisse résoudre correctement les cas futurs. La mise à jour des mesures de maintenance que nous avons associées aux cas quand aucun cas n'est ajouté à la base de cas ainsi que l'initialisation de ces mesures pour les nouveaux cas ajoutés sont effectuées de la même façon que dans CASEP (voir le chapitre 3). La partie de la base de cas non liée aux neurones est prévue pour rendre les cas cible résolus disponibles tout de suite lors de l'utilisation du système, ce qui rend le système évolutif en temps réel. Ceci n'est pas le cas si le réseau de

[16] Comme plusieurs cas peuvent être formés à partir de chaque séquence, les cas recouverts par d'autres cas de la base de cas ne sont pas ajoutés pour éviter d'augmenter la taille de la base de cas inutilement.

neurones est utilisé tout seul.
Les valeurs des différents paramètres du système CASEP2 sont choisies de manière empirique.

7.6 Étude comparative

Le système CASEP2 est un système hybride à composants intégrés où le couplage des modules est étroit puisque les deux module RàPC et RN communiquent par l'intermédiaire d'une mémoire partagée. Dans notre système, les modules RN et RàPC coopèrent au même niveau pour fournir les solutions du cas cible. Le mode d'intégration est celui du co-traitement. De plus, dans le module RàPC, les deux réseaux de neurones effectuent des taches de sous traitement. Le premier réseau M-SOM-ART effectue une indexation de la base de cas tandis que le second fournit la prédiction (ou le classement) dans la phase de réutilisation. Le premier RN permet de modéliser les cas sous forme de matrices de covariance dynamiques et le système RàPC fournit les bases d'apprentissage pour les deux réseaux de neurones.
Dans CASEP2, nous avons utilisé la même structure de mémoire que dans le système ProBIS (Malek, 2000), mais celui-ci ne traite pas les données temporelles et le réseau de neurones utilisé n'est pas le même. De plus l'ajout de cas et l'utilisation de la partie atypique ne sont pas effectués de la même manière. Dans (Corchado et Lees, 2000) et (Fdez-Riverola et Corchado, 2003), les auteurs utilisent les réseaux de neurones dans les différentes phases du cycle de RàPC pour traiter des données temporelles, mais les systèmes proposés prennent en compte l'aspect temporel des données juste en définissant des fenêtres temporelles de longueurs fixées pour représenter les cas, ce qui représente une limite de ces systèmes dans plusieurs applications. Dans CASEP2, aucune restriction n'est imposée sur la longueur de la séquence dans la représentation de cas. Le traitement des données temporelles est effectué par des réseaux de neurones adéquats qui ont les propriétés de stabilité et de plasticité. Celles-ci sont importantes pour une utilisation à long terme du système. De plus, ce réseau est utilisé dans la phase de réutilisation et prend en compte les traitements précédents effectués dans la même séquence. Le réseau de neurones utilisé dans (Fdez-Riverola et Corchado, 2003) pour l'indexation des cas est le GCS, qui est une carte auto-organisatrice évolutive mais qui ne prend pas en compte l'aspect temporel des données et n'assure pas la préservation des anciennes connaissance déjà acquises. L'organisation de la mémoire dans (Fdez-Riverola et Corchado, 2003) n'est pas la même que dans CASEP2 (ceci concerne principalement la partie atypique). Dans les systèmes proposés dans (Corchado et Lees, 2000), un réseau de neurones RBF est utilisé dans la phase de réutilisation mais ne prend pas en compte l'aspect temporel des données.
Concernant la représentation des cas, dans CASEP2, les cas sont représentés par des successions d'instants. Ils constituent des sous-séquences de différentes longueurs. Nous proposons une nouvelle modélisation de séquences, qui prend en compte la distribution des états et leur ordre dans la séquence. Dans CASEP, les cas sont de longueur fixée et la maintenance de la base de cas consiste à associer des mesures aux cas et de réduire la base de cas. Dans CASEP2, les mêmes mesures sont aussi utilisées pour la réduction de la base de cas, mais en plus de cela, cette base

est divisée en plusieurs parties. Ceci permet d'améliorer l'efficacité du système. Dans CASEP, la tache de réutilisation consiste à calculer une confiance associée à chaque classe. Une méthode plus adéquate est utilisée dans CASEP2 puisqu'un réseau M-SOM-ART est utilisé. Celui-ci effectue un classement (ou une prédiction) de séquences et préserve les traitements précédents effectués dans la même séquence.
Dans CASEP2, le système RàPC fournit l'ensemble d'apprentissage pour les deux réseaux de neurones ; permet l'utilisation des nouveaux cas juste après leur ajout à la base de cas (ceci ne peut être effectué si le réseau de neurones est utilisé tout seul) ; et peut fournir des résultats plus précis puisque des cas réels sont utilisés pour résoudre les cas cibles (dans M-SOM-ART, seuls les prototypes sont gardés en mémoire).

7.7 Expérimentations

Nous avons effectué des expérimentations sur des fichiers de traces de navigations (fichiers log) contenus dans le site Web de commerce électronique décrit précédemment (après les différents traitements cités dans le paragraphe §1.2). Le comportement de chaque utilisateur du site est décrit par la succession de pages qu'il a visitées dans le site.
Nous avons conçu et mis en œuvre un prototype du système CASEP2 afin de classer les utilisateurs du site durant leurs navigation dans l'une des deux classes {acheteur, non acheteur}. Ce classement est effectué après chaque page parcourue par l'internaute dans le site[17]. Les expérimentations consistent à comparer les résultats fournis par le prototype du système à ceux fournis par le système CASEP et ceux obtenus en utilisant le réseau M-SOM-ART. Pour comparer la qualité de prédiction de ces deux approches, nous avons utilisé les mesures suivantes (ce sont les mêmes mesures que nous avons utilisées pour valider le système CASEP) :

- *La précision :* représente le taux de bons classements parmi les classements proposés par le système.
- *Le rappel :* représente le taux de bons classements parmi les classements que le système devrait effectuer.
- *Le taux de classements :* représente le taux des classements fournis par le système parmi les classements qu'il devrait effectuer.

Pour évaluer le système CASEP2, nous avons comparé les résultats obtenus dans la phase d'utilisation à ceux obtenus en utilisant CASEP et le réseau M-SOM-ART.
Dans les premières expérimentations, CASEP2 (CASEP2V1 dans les tableaux 7.1, 7.2, 7.3) utilise le module RàPC pour fournir les résultats de classement (les réseaux de neurones sont utilisés seulement pour indexer les cas de la base de cas et dans la phase de réutilisation du système RàPC). Dans les deuxièmes expérimentations, CASEP2 utilise le réseau M-SOM-ART pour fournir les solutions des cas cibles sans utiliser le composant RàPC quand la confiance associée à la classe fournie est plus grande qu'un seuil β. Si cette confiance est inférieure à β, le module RàPC est utilisé (les valeurs de β sont 0.6, 0.7 et 0.8 correspondant à CASEP2V2(0.6), CASEP2V2(0.7)

[17] Les tests effectués pour valider les approches connexionnistes consistent à classer les navigations entières.

et CASEP2V2(0.8) respectivement dans les tableaux 7.1, 7.2, 7.3).
Les résultats sont donnés par les tableaux 7.1, 7.2, 7.3.

	CASEP	M-SOM-ART	CASEP2-V2(0.6)	CASEP2-V2(0.7)	CASEP2-V2(0.8)	CASEP2-V1
Taux de classement	76,62%	100%	100%	100%	100%	100%
Rappel	61%	86,49 %	87,3%	86,82%	84,02%	79,8%
Précision	80%	86,49 %	87,3%	86,82%	84,02%	79,8%

TAB. *7.1: Comparaison des résultats globaux.*

	CASEP	M-SOM-ART	CASEP2-V2(0.6)	CASEP2-V2(0.7)	CASEP2-V2(0.8)	CASEP2-V1
Rappel	43,69%	70,73 %	69,05%	70,33%	75,39%	76,87%
Précision	53,73%	80,76 %	84,77%	82,23%	71,57%	62,60%

TAB. *7.2: Comparaison des résultats obtenus pour la classe acheteur.*

	CASEP	M-SOM-ART	CASEP2-V2(0.6)	CASEP2-V2(0.7)	CASEP2-V2(0.8)	CASEP2-V1
Rappel	70,61%	93,02%	94,86%	93,70%	87,59%	80,97%
Précision	95,38%	88,46%	88,09%	88,40%	89,58%	89,42%

TAB. *7.3: Comparaison des résultats de la classe Non acheteur.*

Le Tableau 7.1 montre que CASEP2V2 (CASEP2V2(0.6) et CASEP2V2(0.7)) donnent les meilleurs résultats globaux que M-SOM-ART et CASEP puisqu'ils utilisent les réseaux M-SOM-ART et le module RàPC pour fournir les solutions des cas cibles. Le résultat global augmente dans CASEP2 avec l'augmentation de l'utilisation du réseau de neurones (quand le seuil β décroît, la fréquence d'utilisation du réseau de neurones croît). CASEP2 et M-SOM-ART fournissent des solutions pour tous les cas cibles, ceci n'est pas le cas pour CASEP. Leur rappel est meilleur que celui de CASEP.
Les tableaux 7.2 et 7.3 montrent que l'augmentation de l'utilisation du réseau de neurones dans CASEP2 augmente la reconnaissance de la classe la plus fréquente (la classe non acheteurs) et l'augmentation de l'utilisation du module RàPC augmente la reconnaissance de la classe rare (la classe acheteur).
Concernant l'efficacité du système, CASEP2 traite les séquences en moins de temps que CASEP (ceci est du au fait que la base de cas est divisée en plusieurs parties et qu'il n'y pas d'extraction

de cas) et le réseau de neurones M-SOM-ART est plus rapide que CASEP2. Dans CASEP2, l'augmentation de la fréquence d'utilisation du réseau de neurones augmente l'efficacité du système.
Ces résultats montrent que le module RàPC arrive à reconnaître les classes rares mieux que le réseau de neurones utilisé tout seul. Celui-ci reconnaît mieux la classe la plus fréquente.

7.8 Conclusion

Nous avons conçu et implémenté un système hybride qui combine le raisonnement à partir de cas avec des réseaux de neurones. Ce système prend en compte l'aspect temporel des données ainsi que leur gros volume et le bruit qu'elles contiennent. Nous avons testé plusieurs variantes de ce système qui diffèrent dans la manière dont les modules sont utilisés. Nos contributions concernent :

- La proposition d'une nouvelle modélisation de cas par des matrice de covariance dynamiques qui représentent les informations contenues dans les séquences ainsi que l'ordre temporel des états contenus dans les séquences. Ce modèle permet, dans une même séquence, de calculer le nouveau modèle du cas cible en fonction du modèle précédent (calcul incrémental).
- L'utilisation d'un nouveau modèle de réseaux de neurones (M-SOM-ART) qui prend en compte l'aspect temporel des données et qui possède les propriétés de plasticité et de stabilité.
- La préservation des traitements effectués dans une même séquence pour résoudre les cas cibles dans la phase de réutilisation.
- La maintenance de la base de cas du système basée sur la combinaison des deux stratégies principales de la maintenance de la base de cas : celle de l'optimisation de la taille de la base de cas et celle de la partition de la base de cas.

Chapitre 8

Conclusion et perspectives

8.1 Bilan

Les contributions que nous avons apportées dans notre travail de thèse peuvent se résumer comme suit :

1. Nous avons proposé un système de RàPC (CASEP) pour le classement ou la prédiction à partir de séquences. Un cas dans ce système est une expérience précise dans une séquence. Pour cela, la mémoire de CASEP est constituée d'une base de cas et d'une base de séquences. Ce système est caractérisé par les points suivants :

   - Les cas ajoutés à la base de cas sont des cas extraits à partir de la base de séquences et non des cas cibles résolus comme dans les systèmes de RàPC classiques. De plus, plusieurs cas peuvent être ajoutés à la base de cas à chaque cycle de RàPC. Les cas sont choisis en utilisant d'abord l'algorithme des matrices de similarité (Fu, 1992) qui forme des groupes de cas, puis en choisissant des représentants parmi les cas qui ont contribué à la résolution du cas cible. Les notions de similarité et de diversité sont prises en compte dans l'ajout de cas pour enrichir la base de cas.

   - Nous avons associé aux cas de la base de cas de nouvelles mesures de maintenance qui permettent de faire face aux problèmes issus du grand volume de données et de la présence du bruit. Ceci est effectué en se basant sur la prise en compte des succès et des échecs lors de l'utilisation des cas concrets pour résoudre les cas cibles.

   - Le seuil de similarité est variable, il est associé à chaque cas de la base de cas et dépend de la qualité du cas basée sur son utilisation pour la résolution des cas cibles. Ceci permet de filtrer une partie du bruit contenu dans les données.

- Nous avons proposé une réduction de la base de cas du système CASEP basée sur les mesures de maintenance associées aux cas.

2. Nous avons ensuite proposé plusieurs nouveaux modèles de cartes auto-organisatrices. Ces modèles ont été construits dans le but de doter ces cartes de certaines propriétés qui sont nécessaires pour notre application. Ces propriétés sont : le traitement des séquences temporelles, la plasticité et la stabilité.

   - Nous avons proposé une nouvelle carte auto-organisatrice évolutive « SOM-ART » qui possède les propriétés de plasticité et de stabilité. Ce modèle utilise le paradigme de la théorie de résonance adaptative pour doter la carte SOM de ces propriétés.

   - Nous avons aussi proposé une nouvelle classe de cartes SOM temporelles « M-SOM », qui modélisent les séquences en utilisant les matrices de covariance. Les premiers modèles, inspirés du domaine de l'identification de locuteurs, ne prennent pas en compte l'ordre temporel dans la séquence. Nous avons alors proposé de nouveaux modèles où la dynamique temporelle est introduite dans la matrice de covariance.
     Dans les approches, où les entrées sont modélisées par des matrices de covariance, nous avons introduit des modifications dans la carte SOM. Ceci est dû au fait que les entrées et les poids des neurones sont des matrices et non des vecteurs.

   - Pour obtenir les propriétés désirées dans un seul modèle, nous avons proposé le modèle « M-SOM-ART » qui combinent les modèles précédemment cités. Il traite les séquence en modélisant les entrées par des matrices de covariance dynamiques et il possède les propriétés de plasticité et de stabilité en se basant sur le paradigme ART.

3. A la fin de cette étude, nous avons travaillé sur la possibilité de coopération des systèmes connexionnistes avec l'approche de RàPC afin de concevoir des systèmes modulaires hybrides. Ces architectures présentent en théorie l'avantage de combiner les points forts de leurs différents composants. Nous avons ainsi proposé un système hybride qui combine le raisonnement à partir de cas avec le réseau M-SOM-ART que nous avons construit. Ce système prend en compte l'aspect temporel des données ainsi que leur gros volume et le bruit qu'elles contiennent. Nous avons testé plusieurs variantes de ce système qui diffèrent dans la manière dont les modules sont utilisés. Dans ce système, nous avons :

   - proposé une nouvelle modélisation de cas par des matrices de covariance dynamiques qui représentent les informations ainsi que l'ordre temporel des états contenus dans les séquences. Ce modèle permet de calculer le nouveau modèle du cas cible en fonction du modèle précédent (calcul incrémental) dans une même séquence.

- utilisé un nouveau modèle de réseaux de neurones (M-SOM-ART) qui prend en compte l'aspect temporel des données et qui possède les propriétés de plasticité et de stabilité.

- pris en compte la préservation d'une trace des différents traitements effectués dans une même séquence pour résoudre les nouveaux cas cibles dans la phase de réutilisation.

- proposé une approche pour la maintenance de la base de cas du système basée sur la combinaison des deux stratégies principales : celle de l'optimisation de la taille de la base de cas et celle de la partition de la base de cas.

La conception du système hybride CASEP2 vise à tirer profit des avantages de chaque technique. D'un coté, le raisonnement à partir de cas permet de rendre le système incrémental en temps réel, ce qui n'est pas le cas du réseau connexionniste qui, malgré le fait qu'il soit évolutif, ne peut être mis à jour que de façon périodique. De plus, le raisonnement à partir de cas permet de fournir des solutions plus précises quand la confiance associée à la solution fournie par le réseau connexionniste n'est pas suffisante puisque celui-ci ne garde en mémoire que les moyennes des exemples tandis que le RàPC garde les exemples concrets dans sa mémoire. Le RàPC permet aussi de fournir une base d'apprentissage pour le réseau connexionniste. D'un autre coté, le réseau connexionniste permet d'indexer la base de cas volumineuse du module de RàPC et de la diviser en plusieurs parties pour limiter l'espace de recherche dans la mémoire. Le réseau connexionniste est aussi utilisé dans la phase de réutilisation du module de RàPC, ceci permet de fournir une solution au problème courant et de garder la trace des raisonnements effectués dans une même séquence.

8.2 Perspectives

Il reste de nombreuses directions de recherche à explorer, nous pensons tout particulièrement aux points suivants :

- Le modèle auto-régressif (AR) (Magrin-Chagnolleau *et al.*, 1996) nous parait très intéressant puisqu'il prend en compte l'ordre des vecteurs dans les séquences par l'utilisation des matrices décalées. Cette approche est considérée comme une manière efficace pour extraire la dynamique des caractéristiques du locuteur. Nous avons commencé à travailler sur ce modèle en utilisant la distance de Frobenius décrite dans le paragraphe 5.2.2, mais nous n'avons pas obtenu de bons résultats pour notre application. Les différentes distances citées dans le paragraphe 5.2.5 seront testées dans plusieurs applications et leurs résultats seront comparés à d'autres approches.

- Les modèles de cartes auto-organisatrices que nous avons proposées effectuent des classement et des classifications de données. Nous nous intéressons à utiliser ces modèles pour effectuer des prédictions. Pour ceci, plusieurs extensions de la carte SOM proposées dans la littérature peuvent être utilisées, parmi lesquelles, nous pouvons citer : la carte SOM étendue (ESOM : Extended SOM) (Ritter et Schulten, 1986), la carte SOM à sortie supervisée (SSOM : Supervised SOM) (Ritter et Schulten, 1988) et la carte SOM utilisant des fonctions à bases radiales (RBF) (Park et Sandberg, 1991).

- Nous nous sommes intéressés dans notre travail particulièrement à l'évaluation des résultats de classements obtenus en utilisant les cartes auto-organisatrices que nous avons proposées puisque c'est le but de notre application. Les résultats de classification ainsi que ceux de visualisation seront étudiés. La propriété de visualisation, qui est l'une des propriétés les plus intéressantes de la carte SOM, va permettre l'étude de l'état de la base de cas, ceci peut conduire à la proposition d'une nouvelle stratégie de maintenance pour le système CASEP2.

- Dans le système CASEP2, la structure des cas est la même dans la composante RàPC et dans le RN. Nous nous intéressons aussi à utiliser une structure plus riche de cas, différente de celle traitée par le réseau de neurones. Ceci pourra affiner les résultats fournis par le système dans des applications où certaines connaissances du domaine sont disponibles.

- Une méthode d'extraction automatique de cas cibles à partir de la séquence courante peut être très utile dans un système de RàPC. Ceci pourra permettre de n'utiliser que les éléments pertinents qui constituent un cas. L'adaptation de certaines méthodes existantes de sélection de variables ainsi que l'utilisation des connaissances *a priori* du domaine peuvent être envisagées pour effectuer cette tâche.

- Les approches connexionnistes et hybrides développées dans cette thèse ont été validées sur un problème réel qui est le traitement de séquences de navigation sur internet. Les approches proposées sont générales et peuvent traiter des données présentées sous forme de séquences multi-variées. Nos contributions sont alors facilement transposables à d'autres problèmes dans d'autres domaines (séquences d'images, séquences biologiques, données spatio-temporelles, données comportementales, etc.)

Annexe A

Nous donnons dans cette annexe les algorithmes détaillés des réseaux ART1 et ART2 issus de la théorie de résonance adaptative (ART) (Grossberg, 1987).

Algorithme ART1 :

Initialisation :

- Constantes :
  $A_1, C_1, D_1 \geq 0$
  $max\{D_1, 1\} < B_1 < D_1 + 1$
  $L > 1$
  $0 < \rho \leq 1$
- Poids $F_2 \Rightarrow F_1(v_j \rightarrow v_i)$: Top-Down
  $w_{ij}(0) > \frac{B_1 - 1}{D_1}$
- Poids $F_2 \Rightarrow F_1(v_i \rightarrow v_j)$: Bottom-Up
  $0 < w_{ij} < \frac{L}{L-1+m}$:
- Les activations sur F_2 :
  $x_{2j}(0) = 0$
- Les activations sur F_1 :
  $x_{1i}(0) = \frac{-B_1}{1+C_1}$

(1) Présenter une forme binaire I sur F_1. Les activations de F_1 sont alors calculées par :
$x_{1i} = \frac{I_i}{1+A_1(I_i+B_1)+C_1}$

(2) Calculer le vecteur S des sorties de F_1 : $s_i = h(x_{1i}) = \begin{cases} 1 \text{ si } x_{1i} > 0 \\ 0 \text{ si } x_{1i} \leq 0 \end{cases}$

(3) Propager S vers F_2 et calculer les activations :
$T_j = \sum_{i=1}^{m} s_i w_{ji}$

(4) Seule la cellule gagnante u_j de F_2 possède une sortie $\neq 0$

$$u_j = f(x_{2j}) = \begin{cases} 1 \text{ si } T_j = \max\limits_k T_k \\ 0 \text{ sinon} \end{cases}$$

(5) Retro-propager la sortie de F_2 vers F_1. Calculer les entrées du réseau à partir de F_2 vers F_1 :
$v_i = \sum_{j=1}^{n} u_j w_{ij}$

(6) Calculer les nouvelles activations :
$x_{1i} = \frac{I_i + D_1 V_i - B_1}{1 + A_1(I_i + D_1 V_i) + C_1}$

(7) Déterminer les nouvelles sorties s_i comme dans (2)

(8) Déterminer le rapport entre la forme I d'entrée et le prototype S :
$\frac{|S|}{|I|} = \frac{\sum_{i=1}^{m} S_i}{\sum_{i=1}^{m} I_i}$

(9) Test de vigilance :
- si $\frac{|S|}{|I|} < \rho$ alors désactiver u_j, annuler les sorties de F_2, aller à 1 en utilisant l'entrée originale.
- sinon, continuer vers (10)

(10) Adaptation des poids de $F_1 \Rightarrow F_2$ (Bottom-up) associés à la cellule v_i :
$$wji = \begin{cases} \frac{L}{L-1+|S|} \text{ si } v_i \textit{ active} \\ 0 \text{ sinon} \end{cases}$$

(11) Adaptation des poids de $F_2 \Rightarrow F_1$ (Top-Down) partant de la cellule v_j vers les unités de F_1 :
$$wij = \begin{cases} 1 \text{ si } v_i \textit{ active} \\ 0 \text{ sinon} \end{cases}$$

(12) Retourner à (1).

Algorithme ART2 :

Initialisation :
- Constantes :
  $a, b > 0$, $0 \leq d \leq 1$, $\frac{cd}{1-d} \leq 1$
  $0 \leq \theta \leq 1$, $0 < \rho \leq 1$, $e << 1$
- Poids $F_2 \Rightarrow F_1(u_j \rightarrow v_i)$: Top-Down
  $w_{ij}(0) = 0$

- Poids $F_2 \Rightarrow F_1(v_i \rightarrow u_j)$: Bottom-Up
  $w_{ij}(0) \leq \frac{1}{(1-d)\sqrt{m}}$

(1) Initialiser les sorties de toutes les sous-couches à 0.

(2) Présenter une forme I sur la sous-couche z.
Les activations de cette sous-couche sont alors calculées par :
$z_i = I_i + au_i$

(3)Propager vers la sous-couche x :
$x_i = \frac{z_i}{e+||z||}$

(4) Propager vers la sous-couche v :
$v_i = f(x_i) + bf(q_i)$

(5) Propager vers la sous-couche u :
$u_i = \frac{v_i}{e+||v||}$

(6) Propager vers la sous-couche p :
$p_i = u_i + dw_{ij}$

(7) Propager vers la sous-couche q :
$q_i = \frac{p_i}{e+||p||}$

(8) Répéter de (2) à (7) jusqu'à stabilisation des valeurs sur F_1.

(9) Calculer les sorties de la couche r :
$r_i = \frac{u_i+cp_i}{e+||u||+c||p||}$

(10) Test de la validité de la forme reconnue par le système :
- Si $\frac{\rho}{e+||r||} > 1$ alors inhiber les cellules actives de F_2 (reset) et aller en (2).
- Sinon
  - si $t = 1$ aller en (11)
  - sinon si $t > 1$ aller en (14)

(11) Propager les sorties de la sous-couche p vers F_2. Calculer les entrées de F_2 :
$T_{ij} = \sum_{i=1}^{m} p_i w_{ij}$

(12) Seule la cellule gagnante de F_2 possède une sortie $\neq 0$:
$$g(T_j) = \begin{cases} d \text{ si } T_j = \max\limits_k T_k \\ 0 \text{ sinon} \end{cases}$$

(13) Répéter les étapes (6) à (10)

(14) Adaptation des poids $F_1 \Rightarrow F_2$ (Bottom-Up) associés à la cellule v_J :
$w_{Ji} = \frac{u_i}{1-d}$

(15) Adaptation des poids $F_2 \Rightarrow F_1$ (Top-Down) partant de la cellule v_J vers les unités de F_1 :
$w_{iJ} = \frac{u_i}{1-d}$

(16) Retourner à (1) avec une nouvelle forme.

Annexe B

L'algorithme initial de SOTPAR proposé dans (Euliano et Principe, 1996) propage l'activité temporelle dans une carte SOM de dimension 1. Nous avons étudié cet algorithme et l'avons généralisé pour une carte de dimension 2. Nous donnons dans cette annexe deux variantes de l'algorithme généralisé de SOTPAR, la première effectue une propagation de l'activité temporelle dans une seule direction de la carte et la deuxième effectue une propagation de l'activité temporelle dans plusieurs directions.

Algorithme SOTPAR avec propagation dans une seule direction

La direction de propagation choisie est celle du voisin du neurone gagnant précédant le plus proche de l'entrée précédente. Comme le voisinage que nous avons choisi est rectangulaire, une des quatre directions ci-dessous est choisie (voir la figure 1).

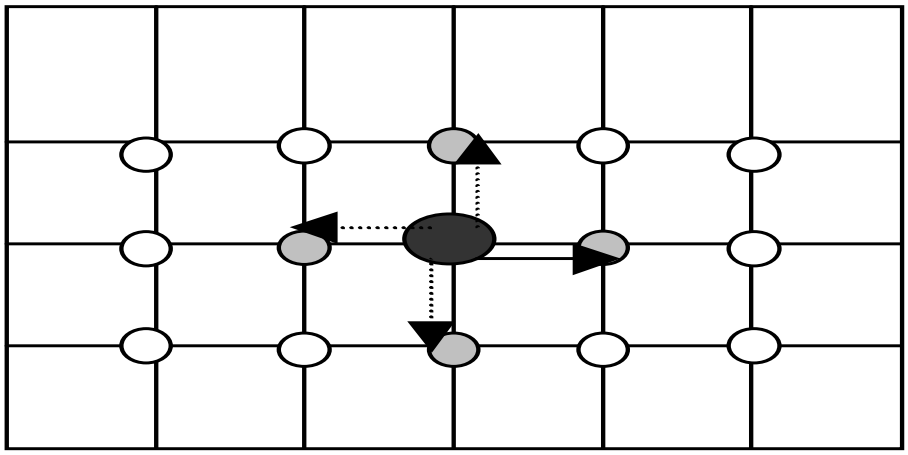


FIG. *1: Les différentes directions possibles pour la propagation de l'activité.*

L'algorithme SOTPAR est décrit ci-dessous :

(a) Initialiser l'ensemble A qui va contenir n unités c_i : $A = \{c_1, c_2, ..., c_n\}$,
les vecteurs $w_i^0 \in R^n$ sont choisis aléatoirement.
Initialiser l'ensemble des connections $M = M1 \times M2$ (voisinage rectangulaire)
Initialiser le paramètre étape d'apprentissage k : $k = 1$.

(b) Pour toute séquence x_k de la base d'apprentissage

1. Initialiser le paramètre temps t : t = 1.
2. Pour tout $i \in \{1, ..., n\}$, $act_i(0) = 0$.
3. Présenter au réseau l'état $x_{t,k}$
4. Calculer les distance entre $x_{t,k}$ et toutes les unités de A.
5. Pour chaque neurone c_i (i = 1, ...,n)
   - Calculer l'activité en c_i au temps (t)

     $$act_i(t) = \alpha * (act_i(t-1) + \delta_{c_i,s(t-1)}) + (1-\alpha) * (act_{vois(c_i)}(t-1) + \delta_{voi(c_i),s(t-1)})$$

     où
     $0 < \alpha \leq 1$ est la constante d'affaiblissement appliquée pour affaiblir l'activité.
     $act_{vois(c_i)}(t)$ est l'activité du voisin de c_i.
     $vois(c_i)$ représente le voisin du neurone c_i qui se trouve dans la direction de propagation.
     δ est la fonction de kroneker.
     $s(t-1)$ est le neurone gagnant à l'instant (t-1).
   - Déterminer le gagnant s(t) : $s(t) = argmin_{c_i \in A}[dist(x_{t,k}, w_i) - \beta * act_i(t)]$
   - Adapter les prototypes :
     $w_i^{nouv} = w_i^{anc} + \mu h_{i,s,k} * dist(x_{t,k}, w_i^{anc})$
     Avec $h_{i,s,k} = exp(\frac{-d_1(c_i,s)^2}{2\sigma^2})$
     $\sigma = \sigma_i(\frac{\sigma_f}{\sigma_i})^{\frac{k}{k_{max}}}$
     $\mu = \mu_i(\frac{\mu_f}{\mu_i})^{\frac{k}{k_{max}}}$
6. FinPour
7. Incrémenter le paramètre du temps : t =t + 1
8. Test de passage à l'état suivant : si $t \leq t_{x_k}$ aller à (c)
   // t_{x_k} représente la longueur de la séquence x_k

(c) FinPour

(d) Incrémenter le nombre d'itérations : k =k + 1

(e) Test d'arrêt : si $k < k_{max}$ aller en 2.

Algorithme SOTPAR avec propagation dans plusieurs directions

Dans cette variante, un voisinage hexagonale est choisi pour représenter la carte. Le sens de propagation étant choisi, l'activité temporelle est propagée à tous les voisins du neurone suivant le sens de propagation (voir la figure 2).

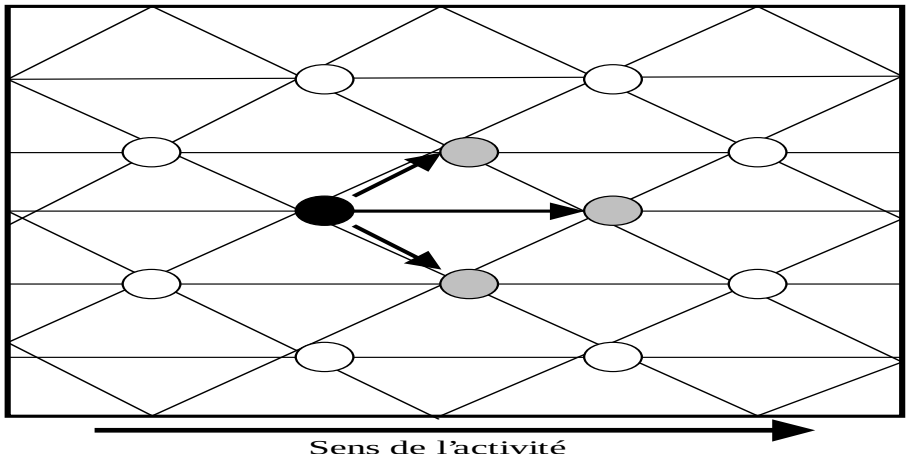


FIG. *2: Les différentes directions pour la propagation de l'activité.*

L'algorithme SOTPAR est décrit ci-dessous :

(a) Initialiser l'ensemble A qui va contenir n unités c_i : $A = \{c_1, c_2, ..., c_n\}$,
les vecteurs $w_i^0 \in R^n$ sont choisis aléatoirement.
Initialiser l'ensemble des connections $M = M1 \times M2$ (voisinage hexagonal)
Initialiser le paramètre étape d'apprentissage k : $k = 1$

(b) Pour toute séquence x_k de la base d'apprentissage

1. Initialiser le paramètre temps t : t = 1
2. Pour tout $i \in \{1, ..., n\}$, $act_i(0) = 0$
3. Présenter au réseau l'état $x_{t,k}$
4. Calculer les distance entre $x_{k,t}$ et toutes les unités de A.
5. Pour chaque neurone c_i (i = 1, ...,n)
   - Calculer l'activité en c_i au temps (t)
     $act_i(t) = \alpha * (act_i(t-1) + \delta_{c_i, s(t-1)}) + (1-\alpha) * \sum_{c_k \in vois(c_i)} (act_{c_k}(t-1) + \delta_{c_k, s(t-1)})$
     où

$0 < \alpha \leq 1$ est la constante d'affaiblissement appliquée pour affaiblir l'activité.

$act_{vois(c_i)}(t)$ est l'activité du voisin de c_i.

$vois(c_i)$ représente les voisins du neurone c_i qui se trouvent dans le sens de propagation (voir la figure 2).

δ est la fonction de kroneker.

- Déterminer le gagnant s : $s = argmin_{c_i \in A}[dist(x_{t,k}, w_i) - \beta * act_i(t)]$
- Adapter les prototypes :
  $w_i^{nouv} = w_i^{anc} + \mu(k-1)h_{i,s,k} * dist(x_{t,k}, w_i^{anc})$
  Avec $h_{i,s,k} = exp(\frac{-d_1(c_i,s)^2}{2\sigma^2})$
  $\sigma = \sigma_i(\frac{\sigma_f}{\sigma_i})^{\frac{k}{k_{max}}}$
  $\mu = \mu_i(\frac{\mu_f}{\mu_i})^{\frac{k}{k_{max}}}$

6. FinPour
7. Incrémenter le nombre paramètre du temps : t = t + 1
8. Test de passage à l'état suivant : si $t \leq t_{x_k}$ aller en 3
   // t_{x_k} représente la longueur de la séquence x_k.

(c) FinPour

(d) Incrémenter le nombre d'itérations : k = k + 1

(e) Test d'arrêt : si $k < k_{max}$ aller en 2.

Annexe C

Nous donnons dans cette annexe les algorithmes d'apprentissage détaillés des modèles de cartes auto-organisatrices évolutives. Ces modèles sont le Growing neural Gas (Fritzke, 1995b), le Growing Cell Structure (Fritzke, 1993) et le Growing Grid (Fritzke, 1995a).

Growing neural Gas

1. Initialisation de l'ensemble des neurones A :

$$A = c_1, c_2 \quad w_{c_1} \in R^n$$

   Initialiser l'ensemble des connexions C,$C \subset A \times A$:

$$C = \emptyset$$

2. Présenter un exemple ξ d'apprentissage
3. Déterminer le premier gagnant s_0, et le deuxième gagnant s_1 :

$$s_0 = \arg\min_{c \in A} ||\xi - w_c|| \quad s_1 = \arg\min_{c \in A - s_0} ||\xi - w_c||$$

4. Si une connexion entre s_0 et s_1 n'existe pas, la créer :

$$C = C \cup (s_0, s-1)$$

$$t_{s_0 s_1} \leftarrow 0$$

5. Incrémenter l'erreur associée à s_0 :

$$\Delta E_{s_1} = ||\xi - w_{s_1}||^2$$

6. Adapter les prototypes :

$$w_{s_0}(t+1) = w_{s_0}(t) + \alpha_b(\xi - w_{s_0}(t))$$

$$w_i(t+1) = w_i(t) + \alpha_n(\xi - w_i(t)) \quad \forall i \in N_{s_0}$$

7. Incrémenter l'âge des connexions partant de s_0 :

$$t_{s_0 i} = t_{s_0 i} + 1, \quad \forall i \in N_{s_0}$$

8. Supprimer les connexions tel que : $t_{s_0 j} > T$
   Supprimer les unités ne possédant pas de connexions
9. Créer une nouvelle unité :
   - déterminer l'unité q possédant la plus grande erreur cumulée :

$$q = \arg\max c \in AE_c$$

   - déterminer parmi les voisins de q, l'unité g avec la plus grande erreur :

$$g = \arg\max c \in N_q E_c$$

   - ajouter une unité r au réseau et initialiser son prototype par :

$$A = A \cup r, \quad w_r = \frac{(w_q + w_g)}{2}$$

   - connecter r à q et g, et supprimer la connexion entre q et g :

$$C = C \cup (r, q), (r, g), \quad C = C\ (q, g)$$

   - Diminuer l'erreur de q et de g d'une fraction δ :

$$\Delta E_q = -\delta E_q, \quad \Delta E_g = -\delta E_g$$

   - approximer l'erreur locale pour l'unité r :

$$E_r = \frac{(E_q + E_g)}{2}$$

10. Diminuer les erreurs locales pour toutes les unités :

$$\Delta E_c = -\gamma E_c, \quad \forall c \in A$$

11. Arrêt si le critère est vérifié (taille du réseau, mesure de performance, ...)

Growing Cell Structure

1. Choisir la dimension k du réseau
   Initialiser l'ensemble A des neurones à $k+1$ unités c_i :

$$A = \{c_1, c_2, ..., c_{k+1}\}$$

   Les poids w_i des neurones c_i sont initialisés de manière aléatoire.
   Initialiser l'ensemble des connexions C, $C \subset A \times A$ de façon que chaque neurone soit connecté à tous les autres.

2. Présenter un exemple ξ d'apprentissage
3. Déterminer le neurone gagnant s :

$$s = \arg\min_{c \in A} ||\xi - w_c||$$

4. Incrémenter l'erreur associée à s :

$$\Delta E_s = ||\xi - w_s||^2$$

5. Adapter les prototypes :

$$w_s(t+1) = w_s(t) + \alpha_b(\xi - w_s(t))$$

$$w_i(t+1) = w_i(t) + \alpha_n(\xi - w_i(t)) \quad \forall i \in N_s$$

où N_s est l'ensemble des voisins de s.

6. Si le nombre des entrées du réseau est un multiple du paramètre λ, insérer une nouvelle unité comme suit :
   - Déterminer l'unité q d'erreur cumulée maximum :

$$q = \arg\max_{c \in A} E_c$$

   - Insérer une nouvelle unité r en divisant la connexion la plus longue liée à q (une connexion entre q et une autre unité f). Insérer les connexions (q, r) et (r, f) et supprimer la connexion (q, f). Pour reconstruire la structure du réseau de façon à avoir des hyper-tetraèdres de dimension k, la nouvelles unité r est aussi connectée à tous les voisins communs entre q et f (avec toutes les unités de $N_g \cap N_f$).
   - Initialiser le vecteur poids du neurone r :

$$w_r = \frac{(w_g + w_f)}{2}$$

   - Diminuer les variable d'erreurs de tous les voisins de r :

$$\Delta E_i = -\frac{\alpha}{|N_r|} E_i \quad (\forall i \in N_r)$$

7. Diminuer les variables d'erreurs de toutes les unités :

$$\Delta E_c = -\beta E_c (\forall c \in A)$$

8. Arrêt si le critère est vérifié (taille du réseau, mesure de performance, ...)

Liste de publications

Conférences internationales et workshops avec comité de lecture

F. Zehraoui, R. Kanawati, S. Salotti. "Case representation, retrieval and reuse in the hybrid system for sequence processing *CASEP2*". Workshop on Applying case-based reasoning to time series prediction. Madrid, Spain, August 2004.

F. Zehraoui, R. Kanawati, S. Salotti. "CASEP2 : Hybrid case-based reasoning system for sequence processing", European Conference on Case Based Reasoning (ECCBR 2004), Madrid, Spain, August 2004.

F. Zehraoui and Y. Bennani, "M-SOM-ART : Growing Self Organizing Map for sequence clustering and classification". European Conference on Artificial Intelligence (ECAI 2004), Valencia, Spain, August 2004.

F. Zehraoui and Y. Bennani, "M-SOM : Matricial Self Organizing Map for sequence clustering and classification", International Joint Conference on Neural Networks (IJCNN 2004), Budapest, Hungary, July 2004.

F. Zehraoui, R. Kanawati, S. Salotti. "Case base maintenance for improving prediction quality". The 6th International Conference on Case-Based Reasoning. 703-717, Trondheim, Norway, June 2003.

F. Zehraoui. "CBR System for sequence prediction CASEP". Workshop on Applying case-based reasoning to time series prediction. 260-269, Trondheim, Norway, June 2003.

Conférences nationales et ateliers avec comité de lecture

F. Zehraoui , R. Kanawati, S. Salotti. "CASEP2 : Système hybride pour le traitement de séquences". 12ème atelier de Raisonnement à Partir de Cas. Villetaneuse, Mars 2004.

F. Zehraoui et Y. Bennani, "SOM-ART : Incorporation des propriétés de plasticité et de

stabilité dans une carte auto-organisatrice", Atelier FDC : Fouille de Données Complexes dans un processus d'extraction de connaissances (EGC 2004), 169-180, Clermont-Ferrand, Janvier 2004.

F. Zehraoui."Améliorer la qualité de prédiction d'un système de raisonnement à partir de cas". 6èmes Rencontres Nationales des Jeunes Chercheurs en Intelligence Artificielle (RJCIA 2003). Juillet 2003, Laval, France, Juillet 2003.

F. Zehraoui, R. Kanawati, S. Salotti."Nouvelle stratégie pour la maintenance de la base de cas".10ème atelier de Raisonnement à Partir de Cas. Paris, Mai 2002.

F. Zehraoui, S. Fabre, R. Kanawati, M. Malek, S. Salotti, E.Janvier."Raisonnement à partir de cas pour la prédiction à partir de bases de données volumineuses constituées de séquences". EGC'02, Extraction des connaissances et apprentissage, Volume 1(4/2001) Hermès, 325-330, Montpellier, Janvier 2002.

Bibliographie

[Aamodt et Plaza, 1994] A. Aamodt et E. Plaza. Case-based reasoning : Fondational issues, methodological variation, and systems. *AI Communication*, 7(1) :36–59, 1994.

[Allen, 1983] J.F. Allen. Maintaining knowledge about temporal intervals. In *Communications of the ACM26*, volume 11, pages 832–843. The University of Rochester, November 1983.

[Anouar, 1996] F. Anouar. *Modélisation probabiliste des cartes auto-organisées : Application en Classification et en Régression*. PhD thesis, CNAM, 1996.

[Baldi *et al.*, 1999] P. Baldi, S. Brunak, P. Frasconi, G. Pollastri, et G. Soda. Exploiting the past and the future in protein secondary structure prediction. In *Bioinformatics*, numéro 15, pages 937–946, 1999.

[Baraldi et Alpaydin, 1998] A. Baraldi et E. Alpaydin. Simplified ART : A new class of ART algorithms. Rapport Technique TR-98-004, Berkeley, CA, 1998.

[Barreto et Arajo, 2001] G.A. Barreto et A.F.R. Arajo. Time in self-organizing maps : An overview of models. *International Journal of Computer Research, Special Issue on Neural Networks : Past, Present and Future*, 10(2) :139–179, 2001.

[Bauer et Villmann, 1997] H.-U. Bauer et T. Villmann. Growing a hypercubical output space in a self-organizing feature map. *IEEE Transactions on Neural Networks*, 8(2) :218–226, March 1997.

[Bellman, 1957] R. Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957.

[Benabdeslem *et al.*, 2002] K. Benabdeslem, Y. Bennani, et E. Janvier. Visualisation and analysis of web navigation data. In *ICANN'02*, pages 486–491, Spain, August 2002. LNCS2415(Springer).

[Bennani *et al.*, 1990] Y. Bennani, F. Fogelman Soulie, et P. Gallinari. A connexionist approach for automatic speaker identification. In *International Conference on Acoustics, Speech and Signal Processing "ICASSP"*, pages 265–272, Albuquerque, New Mexico, USA, April 1990.

[Bennani, 1992] Y. Bennani. *Approches Connexionnistes Pour La Reconnaissance Automatique du Locuteur : Modélisation et Identification*. PhD thesis, Université Paris 11, Orsay, Janvier 1992.

[B.Hammer *et al.*, 2004] B.Hammer, A.Micheli, M.Strickert, et A.Sperduti. A general framework for unsupervised processing of structured data. *Neurocomputing*, 57 :3–35, March 2004.

[Bimbot et Mathan, 1993] F. Bimbot et L. Mathan. Text-free speaker recognition using an arithmetic harmonic sphericity measure. In *Eurospeech'93*, volume 1, pages 169–172, Berlin, Germany, 1993.

[Blackmore et Miikkulainen, 1993] J. Blackmore et R. Miikkulainen. Incremental grid growing : Encoding high-dimensional structure into a two-dimensional feature map. In *IEEE International Conference on Neural Networks (ICNN'93)*, volume 1, pages 450–455, San Francisco, CA, USA, 1993.

[Boyer *et al.*, 1987] A. Boyer, J.P. Haton, et J. di Martino. Dynamic time warping and vector quantization in isolated and connected word recognition. In *European Conference on Speech Technology*, pages 436–439, Edinburgh, GB, 1987.

[Carpenter *et al.*, 1991] G.A. Carpenter, S. Grossberg, et D.B. Rosen. Fuzzy art : Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4 :759–771, 1991.

[Carpenter et Grossberg, 1987] G.A. Carpenter et S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, 37(1) :54–115, janvier 1987.

[Carpenter et Grossberg, 1988] G.A. Carpenter et S. Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *IEEE Computer*, 21(3) :77–88, march 1988.

[Carpenter, 1997] G.A. Carpenter. Distributed learning, recognition, and prediction by art and artmap neural networks. *Neural Networks*, 10(8) :1473–1494, 1997.

[Chappelier *et al.*, 2000] J.C. Chappelier, M. Gori, et A. Grumbach. Time in connexionist models. In *Sequence Learning : Paradigms, Algorithms and Applications*, éditeurs R. Sun et G. Lee Giles, pages 105–134. Springer-Verlag, November 2000.

[Chappell et Taylor, 1993] G.J. Chappell et J.G. Taylor. The temporal kohonen map. *Neural Networks*, 6 :441–445, 1993.

[Corchado et Lees, 2000] J.M. Corchado et B. Lees. Adaptation of cases for case based forecasting with neural network support. In *Soft Computing in Case Based Reasoning*, éditeurs Tharam S. Dillon Sankar K. Pal et Daniel S. Yeung, pages 293–319, March 2000.

[Corvaisier *et al.*, 1997] S.F. Corvaisier, A. Mille, et J.M. Pinon. Information retrieval on the world wide web using a decision making system. *Actes de la* 5^{eme} *conférence sur la Recherche d'Informations Assistée par Ordinateur sur Internet (RIAO'97), Centre des hautes études internationales d'Informatique, Montréal*, pages 285–295, 1997.

[Durand et Alexandre, 1995] S. Durand et F. Alexandre. Spatio-temporal mask learning : Application to speech recognition. In *Proceedings International Conference on Artificial Neural Networks and Genetic Algorithms*, Alès, Avril 1995.

[Elman, 1990] J.L. Elman. Finding structure in time. *Cognitive Science*, 14 :179–211, 1990.

[Euliano et Principe, 1996] N.R. Euliano et J.C. Principe. Spatio-temporal self-organising feature maps. In *Proceedings of the ICNN '96*, volume 1, pages 1900–1905, Washington DC, June 1996.

[Euliano, 1998] N.R. Euliano. *Temporal Self-Organization for Neural Networks*. Dissertation, University of Florida, 1998.

[Fdez-Riverola et Corchado, 2003] F. Fdez-Riverola et J.M. Corchado. Using instance-based reasoning systems for changing environments forecasting. In *Workshop on Applying case-based reasoning to time series prediction*, pages 219–228, Trondheim, Norway, June 2003.

[Fritzke, 1993] B. Fritzke. Growing cell structures – a self-organizing network for unsupervised and supervised learning. TR-93-026, International Computer Science Institute, Berkeley, 1993.

[Fritzke, 1994] B. Fritzke. Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9) :1441–1460, 1994.

[Fritzke, 1995a] B. Fritzke. Growing grid - a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5) :9–13, 1995.

[Fritzke, 1995b] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems*, éditeurs In G. Tesauro, D. S. Touretzky, et T. K. Leen, volume 7, pages 625–632, Cambridge MA, 1995.

[Fu, 1992] G. Fu. An algorithm for computing the transitive closure of a similarity matrix. *Fuzzy Sets and Systems*, 51 :189–194, 1992.

[Fuchs *et al.*, 1995] B. Fuchs, A. Mille, et B. Chiron. Operator decision aiding by adaptation of supervision strategies. In *Case-Based Reasoning : Research and Development (ICCBR'95)*, volume 1010, pages 23–32, 1995.

[Fuchs *et al.*, 1999] B. Fuchs, J. Lieber, A. Mille, et A. Napoli. Towards a unified theory of adaptation in case-based reasoning. In *Case Based Reasonning Research and Development, Third International Conference on Case-Based Reasoning*, pages 104–117, Seeon Monastery, Germany, July 1999. Lecture Notes in Artificial Intelligence 1650, Springer-Verlag, Berlin, Germany.

[Grossberg, 1987] S Grossberg. Competitive learning : From interactive activation to adaptive resonance. *Cognitive Science*, 11(1) :23–63, jan - mar 1987.

[Hagenbuchner *et al.*, 2003] M. Hagenbuchner, A. Sperduti, et A.C. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14 :491 –505, May 2003.

[Hilario *et al.*, 1994] M. Hilario, C. Pellegrini, et F. Alexandre. Modular integration of connectionist and symbolic processing in knowledge-based systems. In *Proceedings International Symposium on Integrating Knowledge and Neural Heuristics*, Pensacola Beach,Florida, USA, May 1994.

[Hilario *et al.*, 1995] M. Hilario, Y. Lallement, et F. Alexandre. Neurosymbolic integration : Unified versus hybrid approaches. In *The European Symposium On Artificial Neural Networks*, Brussels, Belgium, 1995.

[Hilario, 1997] M. Hilario. *An overview of strategies for neurosymbolic integration. In Connectioonist Symbolic integration*, chapitre 1. Lawrence Erlbaum, Hillsdale, NJ, 1997.

[Hochreiter et Schmidhuber, 1996] S. Hochreiter et J. Schmidhuber. Bridging long time lags by weight guessing and "long short-term memory". In *Frontiers in Artificial Intelligence and Applications*, éditeur L. B. Almeida F. L. Silva, J. C. Principe, volume 37, pages 65–72, Amsterdam, Netherlands, 1996. IOS Press.

[Jaczynski et Trousse, 1998] M. Jaczynski et B. Trousse. *www* assisted browsing by reusing past navigations of a group of users. *Proceedings of EWCBR'98, LNAI*, 1488 :160–171, 1998.

[Jaczynski, 1998] M. Jaczynski. *Modèle et plate-forme à objets pour l'indexation par situations comportementales : application à l'assistance à la navigation sur le Web.* Thèse de doctorat, Université de Nice-Sophia Antipolis, 1998.

[Jaere *et al.*, 2002] M D. Jaere, A. Aamodt, et P. Skalle. Representing temporal knowledge for case-based prediction. *ECCBR'02*, 2002.

[James et Miikkulainen, 1995] D.L. James et R. Miikkulainen. Sardnet : A self-organising feature map for sequences. In *Advances in Neural Processing Systems*, éditeurs D.S. Tourestzky G. Tesauro et T.K. Leen, volume 7, pages 577–584, Cambridge, 1995. MA : MIT Press.

[Jha *et al.*, 1999] G. Jha, S.C. Hui, et S. Foo. A hybrid case-based reasoning and neural network approach to online intelligent fault diagnosis. In *Pro.* 3^{rd} *International ICSC Symposia on Intelligent Industrial Automation (IIA'99) and Soft Computing (SOCO'99)*, pages 376–381, Genoa, Italy, 1999.

[Jodouin, 1994] J.F. Jodouin. *Les réseaux Neuromimétiques : Modèles et Applications.* Hermes, 1994.

[Jordan, 1986] M.I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Ninth Annual conference of the Cognitive Science Society*, pages 531–546. Lawrence Erlbaum, 1986.

[Kanawati *et al.*, 1999] R. Kanawati, M. Jaczynski, B. Trousse, et J.M. Andreoli. Applying the broadway recommendation computation approach for implementing a query refinement service in the *cbkb* meta search engine. *Actes de RàPC'99 Raisonnement á Partir de Cas, Plaiseau*, pages 17–26, 1999.

[Kangas, 1991] J. Kangas. Phoneme recognition using time-dependent versions of self-organizing maps. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP'91)*, éditeur IEEE, volume 1, pages 101–104, 1991.

[Kohonen, 1992] T. Kohonen. The hypermap architecture. In *Artificial Neural Network*, éditeurs O. Simula T. Kohonen, K. Makiasara et J. Kangas, volume 1, pages 1357–1360, Amstdam, Netherlands : North-Holland, 1992.

[Kohonen, 1995] T. Kohonen. *Self-Organizing Maps*, volume 30 de *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).

[Lampinen et Oja, 1989] J. Lampinen et E. Oja. Self-organizing maps for spatial and temporal AR models. In *Proc. 6 SCIA, Scand. Conf. on Image Analysis*, éditeurs Matti Pietikäinen et Juha Röning, pages 120–127, Helsinki, Finland, 1989. Suomen Hahmontunnistustutkimuksen seura r.y.

[Leake et Wilson, 1998] D.B. Leake et D.C. Wilson. Categorizing maintenance : Dimensions and directions advances in case-based reasoning. *EWCBR-98*, pages 196–207, 1998.

[Leake et Wilson, 1999] D.B. Leake et D.C. Wilson. When experience is wrong : Examining cbr for changing tasks and environments. *Proceedings of the third International Conference on Case-Based Reasoning, ICCBR-99, Springer-Verlag*, 1999.

[Leake et Wilson, 2000] D.B. Leake et D.C. Wilson. Remembering why to remember : Performance-guided case-base maintenance. *Advances in Case-Based Reasoning : Proceeding of EWCBR-2K, Springer-Verlag*, 2000.

[Leake et Wilson, 2002] D.B. Leake et D.C. Wilson. Maintaining case-base reasoners : dimensions and directions. *Computational Intelligence. Blackwell, in press*, 27(2), 2002.

[Lieber *et al.*, 2004] J. Lieber, M. d'Aquin, S. Brachais, et A. Napoli. Une étude comparative de quelques travaux sur l'acquisition des connaissances d'adaptation pour le raisonnement à partir de cas. In 12^{eme} *Atelier de Raisonnement à Partir de Cas (RàPC 2004)*, pages 53–60, Villetaneuse, France, Mars 2004.

[Magrin-Chagnolleau *et al.*, 1996] I. Magrin-Chagnolleau, J. Wilke, et F. Bimbot. A further investigation on AR-vector models for text-independent speaker identification. In *Proc. ICASSP '96*, pages 101–104, Atlanta, GA, 1996.

[Magrin-Chagnolleau, 1997] I. Magrin-Chagnolleau. *Approches statistiques et filtrage vectoriel de trajectoires spéctrales pour l'identificateur du locuteur indépendante du texte.* PhD thesis, ENST, Janvier 1997.

[Main *et al.*, 2000] J. Main, T.S. Dillon, et S.C.K. Shiu. A tutorial on case based reasoning. *Soft Computing in Case Based Reasoning*, pages "1–28", March 2000.

[Malek et Kanawati, 2001] M. Malek et R. Kanawati. Cobra : A cbr-based approach for predicting users actions in web site. 3^{rd} *International Conference on Case-based Resoning ICCBR'01.*, pages 336–346, 2001.

[Malek, 1996] M. Malek. *Un modèle hybride de mémoire pour le raisonnement à partir de cas.* Thèse de doctorat, Université J. Fourrier, 1996.

[Malek, 2000] M. Malek. Hybrid approaches for integrating neural networks and case based reasoning : From loosely coupled to tightly coupled models. In *Soft Computing in Case Based Reasoning*, éditeurs Tharam S. Dillon Sankar K. Pal et Daniel S. Yeung, pages 73–94, March 2000.

[McCulloch et Pitts, 1943] W. McCulloch et W. Pitts. A logical calculus of the ideas immanent in nervous activity. In *Bulletin of Mathematical Biophysics*, volume 5, pages 115–137. 1943.

[McDermott, 1990] E. McDermott. Lvq3 for phoneme recognition. In *Spring Meet. Acoust. Soc. Jpn.*, pages 151–152, Genoa, Italy, 1990.

[McGarry *et al.*, 1999] K. McGarry, S. Wermter, et J. MacIntyre. Hybrid neural systems : From simple coupling to fully integrated neural networks. *Neural Computing Surveys*, 2 :62–93, 1999.

[McKenna et Smyth, 1999] E. McKenna et B. Smyth. How does your cas base grow ? In *Proceedings of the* 10^{th} *Irish Conference on Artificial Intelligence and Congnitive Science*, Cork, Ireland, September 1999.

[McQueen *et al.*, 2002] T.A. McQueen, A.A. Hopgood, J.A. Tepper, et T.J Allen. A recurrent self-organizing map for temporal sequence processing. In 4^th *Int. Conf. on Recent Advances in Soft Computing (RASC2002)*, pages 43–49, Nottingham, December 2002.

[McSherry, 2002] D. McSherry. A generalised approach to similarity-based retrieval in recommender systems. *Artificial Intelligence Review*, 18 :309–341, 2002.

[Milgram, 1993] M. Milgram. *Reconnaissance des formes : méthodes numériques et connexionnistes*. collection 2ai, 1993.

[M.Strickert et B.Hammer, 2004] M.Strickert et B.Hammer. Self-organizing context learning. In *European Symposium at Artificial Neural Networks'2004*, éditeur M.Verleysen, pages 39–44, 2004.

[Mujica et Vehi, 2003] L.E. Mujica et J. Vehi. Damage identification using case based reasoning and self organizing maps. In *Third European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems*, Oulu, Finland, July 2003.

[Murray-Smith et Thakar, 1993] R. Murray-Smith et S. Thakar. Combining case based reasoning with neural networks. In *AAAI'93 Workshop on AI in Service and Support*, Washington DC, July 1993.

[Needleman et Wunsch, 1970] S.B. Needleman et C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48 :443–453, 1970.

[Park et Sandberg, 1991] J. Park et I.W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2) :246–257, 1991.

[Portinale *et al.*, 1999] L. Portinale, P. Torasso, et P. Tavano. Speed-up, quality and competence in multi-modal case-based reasoning. *ICCBR-99*, pages 303–317, 1999.

[Privitera et Shastri, 1996] C.M. Privitera et L. Shastri. Temporal compositional processing by a dsom hierarchical model. In *International Conference on Artificial Neural Network (ICANN'96)*, pages 457–462, Bochum, Germany, 1996.

[Racine et Yang, 1996] K. Racine et Q. Yang. On the consistency management of large case bases : the case for validation. *American Association for Artificial Intelligence (AAAI-96)*, 1996.

[Racine et Yang, 1997] K. Racine et Q. Yang. Maintaining unstructured case. *Proceeding of the second International Conference on Case Bases Reasoning, RI, USA*, pages 553–564, 1997.

[Ram et Santamaria, 1997] A. Ram et J.C. Santamaria. Continuous case-based reasoning. *Artificial Intelligence*, 90(1-2) :25–77, 1997.

[R.B. Sovat, 2001] A.C.P.L.F. de Carvalho R.B. Sovat. Retrieval and adaptation of cases using an artificial neural network. In *Workshop on Soft Computing in Case-Based Reasoning (in ICCBR'01)*. Vancouver, Canada, July 2001.

[Redmond, 1990] M.A. Redmond. Distributed case for case-based reasoning : Facilitating use of multiple cases. In *Proceedings of the AAAI National Conference on Artificial Intelligence (AAAI'90)*, pages 304–309. MIT press, 1990.

[Reinartz *et al.*, 2000] T. Reinartz, I. Iglezakis, et T. Roth-Berghofer. On quality measure for case base maintenance. *Lecture Note in Artificial Intelligence 1898. Advances in Case-Based reasoning. 5^{th} European Workshop, EWCBR 2000*, pages 247–259, 2000.

[Ritter et Schulten, 1986] H. Ritter et K. Schulten. On the stationary state of kohonen's self-organizing sensory mapping. In *Biol. Cybern.*, volume 54, pages 99–106, 1986.

[Ritter et Schulten, 1988] H. Ritter et K. Schulten. Extending kohonen's self-organizing mapping algorithm to learn ballistic movements. In *Neural Computers*, éditeurs R. Eckmiller et Ch. von der Malsburg, volume 41, pages 393–406. Springer-Verlag, 1988.

[Roh *et al.*, 2003] T.H. Roh, K.J. Oh, et I. Han. The collaborative filtering recommendation based on som cluster-indexing cbr. In *Expert Systems with Applications*, volume 25, pages 413–423. Elsevier Science, October 2003.

[Rosenblatt, 1958] F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, (6) :386–408, 1958.

[Roth-Berghofer et Iglezakis, 2001] T. Roth-Berghofer et I. Iglezakis. Six steps in case-based reasoning : Towards a maintenance methodology for case-based systems. In *The 9^{th} Workshop in the course of annual German Workshop on Case-Based Reasoning, GWCBR 2001*, 2001.

[Rougegrez-Loriette, 1994] S. Rougegrez-Loriette. *Prédiction de processus à partir de comportement observé : le système REBECAS*. Thèse de doctorat, Institut Blaise Pascal, 1994.

[Rumelhart *et al.*, 1986] D.E. Rumelhart, G.E. Hinton, et R.J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing : Explorations in the microstructure of cognition*, éditeurs D. E. Rumelhart et J. L. McClelland, volume 1 :Foundations, pages 318–363., Cambridge, Massachusetts, 1986. The MIT Press.

[Shardanand et Maes, 1995] U. Shardanand et P. Maes. Social information filtering : Algorithms for automating 'word of mouth'. In *Proceeding of ACM CHI'95*, pages 210–217, Denver, CO, 1995.

[Shin et Park, 2000] C.K. Shin et A.C. Park. Towards integration of memory based learning and neural networks. In *Soft Computing in Case Based Reasoning*, éditeurs Tharam S. Dillon Sankar K. Pal et Daniel S. Yeung, volume 1, pages 95–114, March 2000.

[Shiu *et al.*, 2000] S.C.K Shiu, C.H. Sun, X.Z. Wang, et C.S. Yeung. Maintaining case based reasoning systems using fuzzy decision trees. *Lecture Note in Ertificial Intelligence 1898. Advances in Case-Based Reasoning.* 5^{th} *European Workshop, EWCBR 2000.*, pages 285–296, Setember 2000.

[Shiu *et al.*, 2001] S.C.K. Shiu, C.H. Sun, X.Z. Wang, et C.S. Yeung. Transfering case knowledge to adaptation knowledge. an approch for case-base maintenance. *Special issue of the journal Computational Intelligence*, 17(2) :295–314, 2001.

[Smyth et Cunningham, 1996] B. Smyth et P. Cunningham. The utility problem analysed : A case based reasoning perspectives. *Advances in Case Based Reasoning. Lecture Note in Artificial Intelligence*, 1168 :392–399, 1996. Springer-Verlag, Berlin Heidelbarg. New York.

[Smyth et Keane, 1995a] B. Smyth et M.T. Keane. Experiments on adaptation-guided retrieval in a case-based design system. In *Case-Based Reasoning : Research and Development (ICCBR'95)*, éditeurs M. Veloso et A. Aamodt. Lecture Notes in AI. Springer-Verlag, 1995.

[Smyth et Keane, 1995b] B. Smyth et M.T. Keane. Remembring to forget : A competence preserving case deletion policy for case-based reasoning systems. *Proceeding of the fourteenth International Joint Conference on Artificial Intelligence, IJCAI-95*, pages 377–382, 1995.

[Smyth et McClave, 2001] B. Smyth et P. McClave. Similarity vs. diversity. 3^{rd} *International Conference on Case-based Resoning ICCBR'01.*, pages 347–361, 2001.

[Smyth et McKenna, 1998] B. Smyth et E. McKenna. Modelling the competence of case-bases. *Advances in Case-Based Reasoning,* 4^{th} *European Workshop, EWCBR-98*, pages 208–220, 1998.

[Smyth et McKenna, 1999] B. Smyth et E. McKenna. Building compact competent case-bases. *Proceeding of the third International Conference on Case-Based Reasoning*, pages 329–342, 1999.

[Smyth et McKenna, 2002a] B. Smyth et E. McKenna. Competence guided incremental footprint. *Journal of Knowledge-Based Systems, In Press*, 2002.

[Smyth et McKenna, 2002b] B. Smyth et E. McKenna. Competence models and the maintenance problem. *Computational Intelligence : Special Issue on Maintaining Case-Based Reasoning Systems, In Press*, 2002.

[Smyth, 2000] B. Smyth. Competence models and their application. In *Lecture Note in Artificial Intelligence 1898. Advances in Case-Based Reasoning.* 5^{th} *European Workshop, EWCBR 2000*, pages 1–2. Trento, Italy, September 2000.

[Somervuo et Kohonen, 1999] P. Somervuo et T. Kohonen. Self-organizing maps and learning vector quantization for feature sequences. In *Neural Processing Letters*, numéro 10(2), pages 151–159, 1999.

[Varsta *et al.*, 1997] M. Varsta, J. Millán, R. del, et J. Heikkonen. A recurrent self organizing map for temporal sequence processing. In *ICANN'97 : International Conference on Artificial Neural Networks*, volume 1327, pages 421–426. Springer, 1997.

[Varsta *et al.*, 2001] M. Varsta, J. Heikkonen, J. Lampinen, et J.del.R. Millan. Temporal kohonen map and the recurrent self-organizing map : Analytical and experimental comparison. In *Neural Processing Letters*, volume 13, pages 237–251. Kluwer Academic Publishers, 2001.

[Voegtlin, 2002] T. Voegtlin. Recursive self organizing maps (rsom). *Neural Networks*, 15(9-8) :979–992, 2002.

[Waibel *et al.*, 1989] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, et K. Lang. Phoneme recognition using time-delay neural networks. *IEEE, Transactions on Acoustics, Speech and Signal Processing*, 37(3), March 1989.

[Wermter et Sun, 2000] S. Wermter et R. Sun. *Hybrid Neural Systems*. Springer-Verlag, Heidelberg, 2000.

[Yang et Wu, 2000] Q. Yang et J. Wu. Keep it simple : A case-base maintenance policy based on clustering and information theory. *Proceeding of Canadian AI Conference. Montreal, Canada*, 2000.

[Yang et Zhu, 2001] Q. Yang et J. Zhu. A case addition policy for case-base maintenance. *Computational Intelligence Journal, A Special Issue on Case-Base Maintenance. Blackwell Publishers, Boston MA UK*, 17(2) :250–262, 2001.

[Zeboulon *et al.*, 2003] A. Zeboulon, Y. Bennani, et K. Benabdeslem. Hybrid connectionist approach for knowledge discovery from web navigation patterns. In *Proceedings of ACS/IEEE International Conference on Computer Systems and Applications*, Tunisia, July 2003.

[Zehraoui *et al.*, 2003] F. Zehraoui, R. Kanawati, et S. Salotti. Case base maintenance for improving prediction quality. In *The* 6^{th} *International Conference on Case-Based Reasoning (ICCBR-2003)*, pages 703–717, Trondheim, Norway, June 2003.

[Zehraoui *et al.*, 2004] F. Zehraoui, R. Kanawati, et S. Salotti. Casep2 : Hybrid case-based reasoning system for sequence processing. In *The* 7^{th} *European Conference on Case-Based Reasoning (ECCBR-2004)*, Madrid, Spain, August 2004.

[Zehraoui et Bennani, 2004a] F. Zehraoui et Y. Bennani. M-som-art : Growing self organizing map for sequences clustering and classification. In 16*th European Conference on Artificial Intelligence (ECAI2004)*, Valancia, Spain, August 2004.

[Zehraoui et Bennani, 2004b] F. Zehraoui et Y. Bennani. M-som : Matricial self organizing map for sequences clustering and classification. In *International Joint Conference on Neural Networks (IJCNN 2004)*, Budapest, Hungary, July 2004.

[Zehraoui et Bennani, 2004c] F. Zehraoui et Y. Bennani. Som-art : Incorporation des propriétés de plasticité et de stabilité dans une carte auto-organisatrice. In *Atelier FDC : Fouille de Données Complexes dans un processus d'extraction de connaissances (EGC 2004)*, pages 169–180, Clermont-Ferrand, Janvier 2004.

[Zehraoui, 2003] F. Zehraoui. Cbr system for sequence prediction "casep". In *Workshop on Applying case-based reasoning to time series prediction*, pages 260–269, Trondheim, Norway, June 2003.