Université Paris – Nord

THESE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE PARIS 13

Présentée par

Khalid BENABDESLEM

Spécialité Informatique

Sujet:

Approches connexionnistes pour la visualisation et la classification des séquences évolutives : Application aux données issues d'usages d'Internet

Le 17 Décembre 2003

Devant le jury composé de

Mme	Sylvie	THIRIA	Présidente
M.	Mohamed	QUAFAFOU	Rapporteur
M.	Dominique	LAURENT	Rapporteur
M.	Younès	BENNANI	Directeur
M.	Philippe	DAGUE	Examinateur
Mme.	Adeline	NAZARENKO	Examinatrice
Mme.	Catherine	RECANATI	Examinatrice
M.	Eric	JANVIER	Invité

Remerciements

J’adresse mes sincères remerciements à monsieur Mohamed Quafafou, Professeur à l’université d’Avignon et monsieur Dominique Laurent, Professeur à l’université de Cergy Pontoise qui ont accepter d’évaluer ce travail.

J’exprime ma profonde gratitude à Madame Sylvie Thiria, Professeur à l’université de Versailles Saint Quentin en Yveline, pour avoir accepté de présider le jury de ma thèse.

Je tiens également à remercier monsieur Philippe Dague, Professeur à l’université Paris 13, madame Adeline Nazarenko, Maître de conférence à l’université Paris 13 et Madame Catherine Récanati, Maître de conférences à l’université Paris 13; pour avoir accepté d’examiner mes travaux.

Je suis très reconnaissant à mon directeur de thèse, monsieur Younès Bennani, Professeur à l’université Paris 13. Je le remercie pour ses précieux conseils et pour son soutien pendant toutes ces années. Cela a été pour moi un véritable plaisir de travailler avec lui. Je n’oublie pas de remercier tous les membre de l’équipe universitaire du laboratoire LIPN.

Je suis aussi très reconnaissant et je tiens à remercier monsieur Eric Janvier, Directeur associé à LDANumSight. Je le remercie pour ses précieux conseils et pour son soutien à l’intérieur de l’entreprise. Sa rigueur de travail et sa gentillesse ont également contribué au bon déroulement de mes recherches au sein de l’entreprise.

Je tiens à exprimer ma sympathie à toutes les personnes avec qui j’ai travaillé durant ces trois années de thèse : Farida Zehraoui, Rushed Kanawati, Sylvie Salloti, Arnaud Zeboulon, Fabrice Bossaert, Antonio Albuquerque, Sandrine Fabre, Emmanuel Ecosse et Frédéric Martin.

Enfin, merci à mes grands parents, mes parents, ma femme, ma belle famille qui m’ont encouragé et soutenu sans arrêt. Je remercie aussi toute ma famille qui m’a soutenu pendant toutes ces années de thèse.

Table des Matières

Table des Figures

Liste des tableaux

Chapitre I

1 Introduction Générale : Problématique

Dans ce chapitre nous allons exposer la problématique du E-Mining (La fouille électronique). Ce dernier est en réalité un processus de modélisation de l'utilisateur, qui consiste à la recherche des connaissances à partir des données issues d'Internet.

1.1 Introduction

La prolifération de l'information dans le Web a rendu nécessaire la personnalisation de l'espace de cette information. Cette personnalisation peut être faite via des mécanismes de traitement d'informations (par exemple : les moteurs de recherches) ou bien d'une manière « *end to end* » en rendant les sites Web adaptatifs. La personnalisation « *end to end* » est basiquement focalisée sur la création d'entités intermédiaires (systèmes à base de requêtes), souvent considérées comme des systèmes de recommandation. Plusieurs formes primitives de ce type de personnalisation peuvent être vues dans des sites qui demandent aux utilisateurs de fournir des informations personnelles (adresse, numéro de téléphone, code zip, mots clés indiquant leurs intérêts …).

Ce type de personnalisation nécessite une acquisition explicite des données. Ces données sont utilisées ensuite dans un modèle de traitement de tâches, pour fournir un résultat sur le profil de l'utilisateur. Cependant, ce type d'acquisition n'est pas satisfaisant du fait que l'utilisateur fournit rarement ses informations personnelles via le web. Pour cette raison, nous proposons de focaliser notre étude dans un cadre d'adaptation d'interface (dans notre cas l'interface d'un site web) pour extraire des profils d'utilisateurs sans leur demander de remplir des formulaires. Ces profils sont exploités par la suite, pour élaborer un processus de prédiction d'intérêts que peuvent avoir les internautes face à un site web.

1.2 Modélisation d'utilisateur

Les recherches en interaction Homme-Machine (IHM) visent à rendre la machine facilement manipulable pour l'utilisateur. Cette facilité peut incorporer des notions d'intuition, de besoins et de plaisir. Cependant, le challenge est clair : il consiste à concevoir des systèmes qui permettent de répondre aux besoins des utilisateurs. Cela nécessite a priori, plusieurs informations sur leurs caractéristiques afin de les modéliser pour pouvoir les comprendre. En d'autres termes, il s'agit d'extraire leurs profils suivant leurs interactions avec l'interface**[37]**.

Pour effectuer une bonne modélisation d'un utilisateur nous avons besoin de plusieurs informations le concernant, pas en terme personnel mais en terme comportemental[1]. Pour réaliser cette

[1] A vrai dire, un système de modélisation nécessite tout type d'informations pouvant décrire un utilisateur. Cependant, un minimum d'information est requis. Cela dit, son comportement doit être étudié suivant sa façon d'interagir avec l'interface.

modélisation, nous sommes amenés à étudier son interaction en construisant un système d'analyse de traces**[44]**.

1.3 Les difficultés rencontrées en modélisation d'utilisateurs des sites Web

En général, n'importe quel problème serait résolu à l'aide d'un formalisme donné, ne serait ce que par son interprétation. Or, le E-Mining présente des caractéristiques qui rendent complexe son interprétation. Cette complexité est due en grande partie au :

- Manque de données pertinentes.
- Variabilité du comportement inter et intra–Internaute.
- Acquisition implicite des données qui incluent beaucoup de bruits.

Il est important de mentionner que plusieurs efforts ont été faits sur la modélisation, via des techniques relativement simples. Ces techniques sont souvent inadéquates au profil réel de l'utilisateur. Ce profil est assez complexe à comprendre. Cette complexité est due à la *contamination* du bruit, qui est souvent inconnu a priori. De plus, des informations incomplètes peuvent facilement apparaître dans l'espace de données. Cela est dû à une large variété de raisons de connexion, d'ouverture de session et de navigation**[07].**

1.4 La reconnaissance des formes en modélisation d'utilisateur

Les informations observées sur les caractéristiques des utilisateurs peuvent être bruitées. Donc, tous les aspects d'amélioration de données doivent être examinés avant la conception de tout système de prise de décision. Dans ce sens, le système a pour objectif d'établir des profils d'utilisateurs complets et consistants à partir d'informations incomplètes et bruitées.

La modélisation d'utilisateur requiert donc, des caractéristiques d'association et de classification de formes. Les réseaux de neurones possèdent ces caractéristiques et peuvent donc être utilisés pour implémenter des modèles d'utilisateurs. De plus ils peuvent être entraînés à générer des inférences**[37]**.

Les informations maintenues dans les exemples d'apprentissage par les réseaux de neurones sont utilisées pour construire une structure de décision. Dans ce cas, une forme est présentée à un système d'apprentissage de la même façon que les exemples entraînés auparavant. Par conséquent, le système retourne la classe d'appartenance de cette forme.

1.5 Les avantages des techniques de la reconnaissance des formes

Traditionnellement, la modélisation d'utilisateur avait employé les techniques basées sur des connaissances. Ces techniques sont puissantes et efficaces mais souffrent de plusieurs inconvénients. En effet, le problème le plus persistant est celui de l'acquisition des connaissances : leur formulation et leur rassemblage sont coûteux en temps de calcul et provoquent éventuellement des erreurs**[37]**.

Afin d'être efficace, la base de connaissances doit être correcte et complète. Si les informations sont corrompues, les inférences seront fausses et si elles ne sont pas disponibles, les inférences ne peuvent même pas être faites**[37].**

Dans la modélisation d'utilisateur, les comportements varient dramatiquement et involontairement. Ce qui rend difficile l'assurance de la complétude et la correction.

Les méthodes de reconnaissance des formes possèdent un certain nombre d'avantages sur les approches à base de connaissances :

- Elles apprennent à partir d'exemples et ne nécessitent pas la formulation explicite de connaissances et de règles. Cela évite le problème d'acquisition et permet aux experts d'identifier les cas représentatifs de chaque type.
- Ces méthodes permettent une généralisation qui réduit la nécessité de la complétude dans la connaissance fournie. Bien qu'ils procèdent de différentes manières, tous les systèmes de reconnaissance de formes reconnaissent des formes non vues auparavant (i.e. qui n'ont pas participé à l'apprentissage). Cela dit, l'utilisation d'un ensemble complet d'exemples n'assure pas nécessairement la performance du système.
- Les classifieurs de formes sont plus portables et plus extensibles. En effet, les exemples peuvent être ajoutés ou bien tout le système peut être réentraîné pour reconnaître de nouvelles caractéristiques. Ces techniques sont aussi flexibles d'être combinées pour fournir des profils sophistiqués composés de plusieurs bases de classification.

1.6 E-Mining

1.6.1 Présentation

Le E-Mining peut être considéré ici comme un processus de modélisation d'utilisateur. En effet, il peut être vu sous toutes les

dimensions de la modélisation. Il en représente donc une application particulière.

- Les utilisateurs : les internautes.
- Type de modélisation : dynamique, analyse de mouvement dans le temps[2].
- Interfaces : adaptative organisée sous la forme de plusieurs connexions de pages.
- Formes : traces de pages formant des sessions.
- Acquisition : explicite par défaut (recherche par mots clés, remplissage de formulaire). Dans notre cas, elle est implicite car elle ne dépend que des clics.

On peut donc utiliser un processus de reconnaissance des formes pour construire des profils d'internautes. Ce processus est appelé : E-Mining.

Le concept du E-Mining s'appuie sur le constat, qu'il existe au sein de chaque site, des informations cachées dans le gisement de données issues de ce site. Ces données permettent – grâce à un certain nombre de techniques spécifiques – de faire apparaître des connaissances.

L'exploration des données se fait à l'initiative du système. Son but est de remplir l'une des tâches suivantes : classement, estimation, prédiction, regroupement par similitudes, segmentation, dans une moindre mesure, l'optimisation. De nombreuses adaptations de ces techniques se font sur différents supports et génèrent un nouveau vocabulaire (Text-Mining, Multimédia-Mining, Web-Mining, …etc.)**[27]**.

1.6.2 Les problèmes traités en E-Mining

Sur certains serveurs de sites, le nombre de connexions quotidiennes peut atteindre des volumes très importants. Il est donc probable de trouver des informations pertinentes, qui nous permettent d'extraire des profils significatifs sur le comportement des internautes. Par exemple, une entreprise ayant un site commercial, peut avoir plusieurs informations sur ses clients, sans avoir la capacité de comprendre leurs comportements (TAB 1.1).

[2] Dans les sites commerciaux, cette analyse concerne les transactions en ligne et l'extraction de connaissance sur les clients

Question	**Technique**
Si les clients achètent le produit A, achèteront t–ils le produit C ou X et quand ?	Séquencement
Qu'est ce qui différencie mes clients fidèles des autres ?	Segmentation
Qui sont mes clients potentiels et comment je les garde ?	Classification
Quel rapport y a t–il entre type de visiteurs et ventes dans le site ?	Association
Comment pourrais-je reconnaître la proportion élevée d'achat ?	Classement
Quels attributs assurent le retour des visiteurs dans mon site ?	Sélection
Comment procéder pour changer le design du site pour maximiser les ventes en ligne ?	Optimisation

TAB 1.1. ***EXEMPLE DE PROBLEMES POSES EN E-MINING***

Il faut savoir que les techniques d'analyse en E–Mining ne sont pas exclusivement attachées entre elles et n'ont pas besoin d'être utilisées séquentiellement. Ce sont les besoins d'applications qui déterminent quand chacune d'elle doit être utilisée.

Similairement à la modélisation d'utilisateur, les outils en E-Mining sont basés sur les techniques d'intelligence artificielle, conçues pour simuler la perception et l'apprentissage humains **[10]**. L'avantage des outils du E-Mining réside dans la façon de traiter les formes de manière autonome : Elles sont conduites par des données contenues dans les bases de données générées par le Web.

Dans cette thèse, nous contribuons dans chaque étape du processus d'extraction des connaissances à partir de données (ECD). En effet, nous proposons plusieurs méthodes de codage pour la préparation des données ; nous proposons une nouvelle heuristique pour le calcul du nombre de neurones dans la carte de Kohonen ; nous modifions cette dernière pour l'adapter aux séquences ; nous développons deux différentes approches pour définir une typologie de sessions et enfin nous élaborons une nouvelle approche qui consiste à rendre la classification évolutive.

1.7 Le plan du rapport

Ce rapport est composé de six chapitres présentés selon l'organisation suivante :

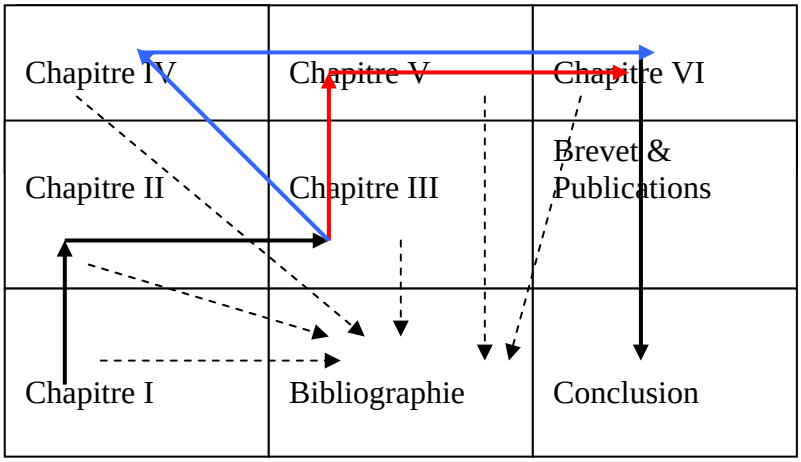


FIG 1.1. ***LE PLAN DU RAPPORT***

- Le présent chapitre était une description du motif de notre étude pour remédier aux difficultés de la modélisation d'utilisateur des sites web. Nous avons ainsi, présenté le lien entre le processus de modélisation d'utilisateur et le E-Mining.
- Dans le deuxième chapitre, nous présenterons la nature des données qui décrivent le comportement des internaute dans un site web. De plus, nous proposerons une série de méthodes de codage nécessaire pour la préparation et le traitement des données.
- Le troisième chapitre sera consacré à l'étude des méthodes de classification non supervisée (carte topologiques de Kohonen, et classification hiérarchique), qui seront les outils utilisés pour les besoins de notre étude. En apportant des modifications sur ces méthodes, nous exposerons une solution de visualisation du plan d'un site web tels que les internautes le perçoivent.
- Dans le chapitre 4, le lecteur pourra comprendre comment les internautes naviguent dans un site web. Pour cela nous décrirons la méthode mise en œuvre pour extraire les différents profils qui décrivent les différents comportements critique des internautes. Cette méthode visera à coder les sessions, qui sont de longueurs différentes, pour appliquer la méthode de classification numérique nécessaire pour la typologie.

- Une autre approche de profiling sera décrite dans le chapitre 5. son but sera de répondre aux limites de la méthode du chapitre 4. Cette approche procède par la modélisation par mélange de chaînes de Markov dont les états seront représentés par les neurones de la carte topologique décrite dans le chapitre 3. La méthode que nous allons développer, fera l'objet d'un processus automatique qui assurera une analyse descriptive des sessions toute en gardant leur nature symbolique bien qu'elle soit floue et difficile à comprendre.
- L'objectif du chapitre 6 est de présenter une version évolutive pour les cartes topologiques de Kohonen. Cette version consiste à rendre évolutif, la découverte de l'espace topologique : i.e. le nombre de regroupements homogènes changera dans le temps en fonction des informations qui arrivent. Une méthode qui répondra donc, au problème d'arrivées des fichiers Log pour le même site web tout en gardant son modèle de cartographie initial.

Chapitre II

2 Données comportementales : Codage

Dans ce chapitre, nous allons présenter l'aspect implicite des connaissances utilisées dans notre étude. Vu la nature complexe de ces connaissances, nous présenterons une série de codages que nous avons développés pour structurer et formater nos données dites comportementales.

2.1 Introduction

A vrai dire, un processus de E-mining complet nécessite plusieurs types de données. Or dans notre étude nous n'utilisons que le type comportemental pour des raisons de coût et de disponibilité. Cela a l'avantage de se focaliser sur le travail de l'intelligence du système. En d'autres termes, notre travail consiste à rendre le système autonome et capable de réagir aux données implicites sans gêner l'utilisateur en lui demandant de remplir des formulaires

FIG 2.1 ci-dessous montre l'éventuelle collaboration de plusieurs types de données pour le E-Mining. Or, seule les données comportementales sont utilisées pour une raison de coût et de disponibilités. Ces données sont utilisées dans une chaîne de traitements pour concevoir un modèle automatique.

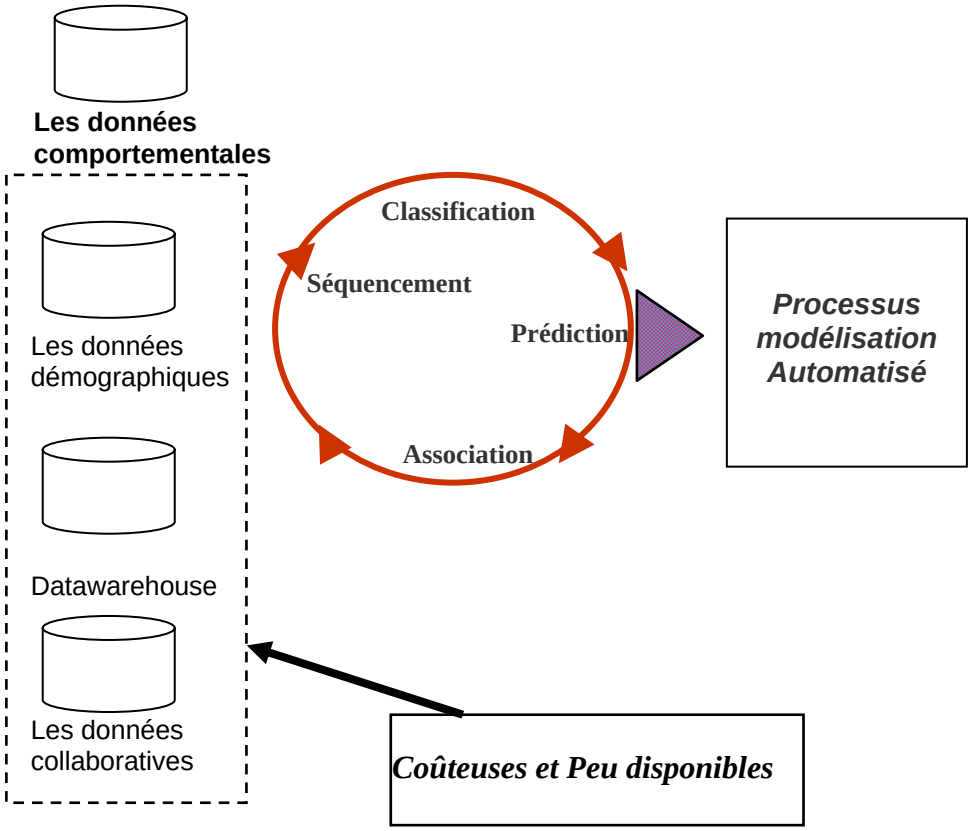


FIG 2.1. ***ARCHITECTURE GENERALE DU PROCESSUS DE MODELISATION D'INTERNAUTES***

2.2 Les données comportementales

Quand un utilisateur navigue dans un site Web, il génère plusieurs transactions par les requêtes qu'il formule durant sa navigation. Il peut accéder aux différentes pages par des hyper – liens et via la recherche par mots clés. Ses traces sont enregistrées dans des fichiers générées par le serveur du site. Un fichier contient en général, des enregistrements des transactions des pages demandées (ex : index.asp, home.htm), les images qui vont avec (ex : bouton.gif, backgroug.jpg) et pleins d'autres fichiers. Presque tous les serveurs enregistrent les fichiers de transactions sous format texte et incorporent au minimum deux types de fichiers : un fichier qui informe sur ce que voit le visiteur du site et un fichier erreur qui donne des informations sur une éventuelle déconnexion. Le premier type de fichier contient toutes les transactions des visiteurs du site, et le cheminement de ces transactions représente l'aspect comportemental des données. Ceci dit, ces dernières sont dites comportementales.

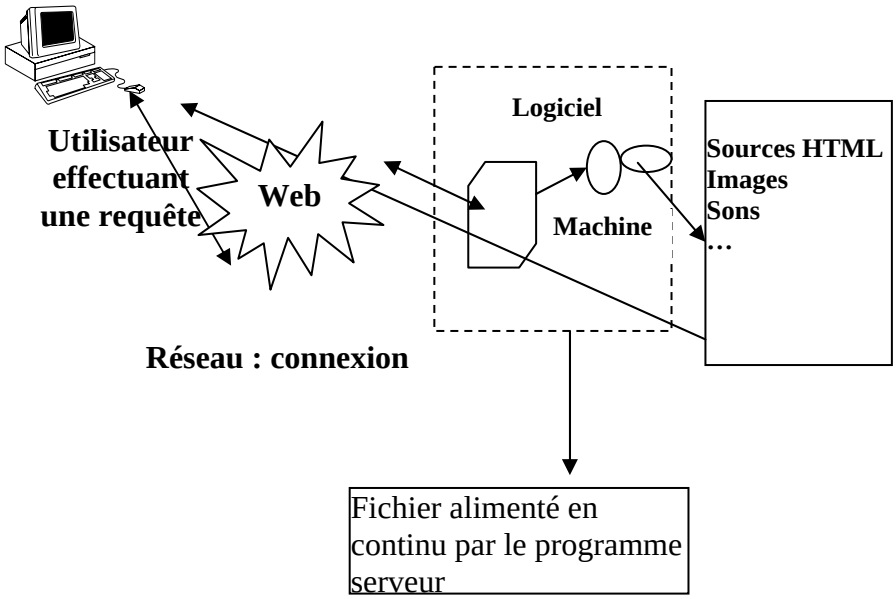


FIG 2.2. ***LE MODE D'EXTRACTION D'INFORMATIONS DU SERVEUR D'UN SITE.***

La figure FIG 2.2 illustre la génération d'informations par le serveur d'un site en fonction des navigations effectuées des utilisateurs dans ce site.

2.3 Le fichier Log

Une des importantes sources d'informations sur la navigation dans un site Web est le fichier Log. Connu aussi sous le nom du « Accès Log » où sont enregistrées toutes les transactions entre le serveur et le navigateur.

Il existe plusieurs formats de représentation d'un fichier Log. La méthode standard par laquelle les serveurs enregistrent les accès par les navigateurs est intitulée : « *Common Log Applications* » créée par NCSA (National Center for Supercomputing Applications). Chaque enregistrement dans un fichier Log est caractérisé par les champs suivants :

- Le temps (time): il représente l'heure de la demande de la page.
- L'adresse IP(c-Ip) : IP est le protocole de transfert des données sur internet.
- L'identificateur de l'internaute (cs-username) : c'est un attribut décrivant l'identité de l'utilisateur.
- L'URL (cs-Url) : c'est l'adresse de la page visité par l'internaute.
- La méthode de la requête (cs-method) : ce champ contient la commande « GET » qui informe le serveur sur le document sur lequel le navigateur lance sa requête. Il existe deux autres commandes : « POST » qui apparaît quand le bouton de soumission est pressé. Cette commande permet au serveur de demander quelques données sur le script qui peut être utilisé. Une troisième commande peu fréquente est la commande « HEAD » qui a la même fonction que « GET ». Sauf que le serveur retourne seulement l'entête du document,
- Le statut (sc-status) : ce champ décrit l'état de la transaction par un code conventionnel. Cela explique simplement que le serveur délivre les pages sélectionnées par le navigateur avec succès.

Le tableau suivant représente différentes classes de codes :

Code	Etat
200	Succès
300	Redirection
400	Défaillance
500	Erreur serveur

TAB 2.1. ***LES DIFFERENTS CODES DECRIVANT L'ETAT DE LA TRANSACTION***

- La taille du fichier reçu (cs-bytes): elle représente le nombre total d'octets transférés par le serveur au client durant la transaction,
- L'agent (cs-agent) : ce champs contient le nom et la version du navigateur,
- Le référent (cs-referer): le dernier champ contient l'URL origine de la requête. Il peut également définir un lien à partir d'une autre page ou un moteur de recherche.

#Software: Microsoft Internet Information Server 4.0
#Version: 1.0
#Date: 2000-06-13 00:00:22

Time	c-ip	cs-username	cs-method	cs-Url	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.wstore.fr/Navigation/searchack.asp?action=search%2FperCategory%2Easp&cat_id=524
00:02:00	213.36.9.210	-	GET	/images/FiletGreenL.gif	200	234	Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98)	http://www.wstore.fr/Navigation/searchack.asp?action=search%2FperCategory%2Easp&cat_id=524
00:02:02	213.36.9.210	-	GET	/images/FiletGreenR.gif	200	400	Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98)	http://www.wstore.fr/Navigation/searchack.asp?action=search%2FperCategory%2Easp&cat_id=524
00:02:02	213.36.9.210	-	GET	/images/BkgdFiletGreen.gif	200	420	Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+98)	http://www.wstore.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)	-

TAB 2.2. ***EXEMPLE D'UN FICHIER LOG A L'ETAT BRUT.***

Cs indique une action client vers serveur, ***sc*** indique une action serveur vers client, ***c*** indique une action client et ***s*** indique une action serveur.

Exemple : 1er enregistrement dans TAB 2.2

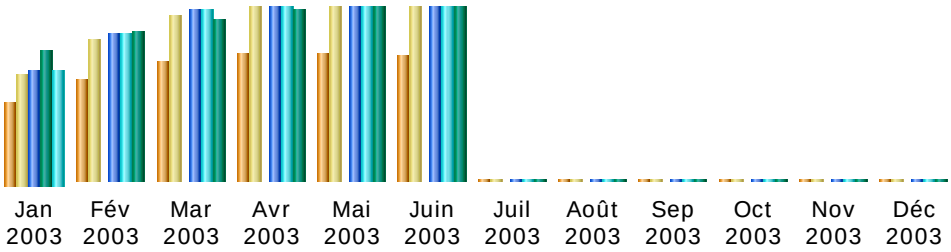
Le 13/06/2000 à 01h39, l'utilisateur inconnu (-) a utilisé la machine 213.36.9.210 pour télécharger la page /index.asp de taille 245 sans erreurs.

2.4 Exemple d'analyse statistique d'un site web

Presque tous les systèmes d'analyse de site Web offrent des rapports et des graphiques basés sur des statistiques de base. A titre d'exemple, nous avons choisi de présenter une analyse descriptive d'un site web (www.awstat.net) pour illustrer les facteurs préliminaires pouvant éventuellement aider à choisir l'analyse de Data Mining à mettre en œuvre.

Résumé

Visiteurs différents	Visites	Pages	Hits	Bande passante
31838	**44626** (1.4 visites/visiteur)	**68462** (1.53 pages/visite)	**68462** (1.53 hits/visite)	**1.94 Go** (45.6 Ko/visite)



Mois	Visiteurs différents	Visites	Pages	Hits	Bande passante
Jan 2003	21060	28083	44398	44398	1.51 Go
Fév 2003	25902	35755	56627	56627	1.66 Go
Mar 2003	30013	42195	65518	65518	1.79 Go
Avr 2003	32046	44685	68114	68114	1.90 Go
Mai 2003	32191	45768	69093	69093	2.01 Go
Juin 2003	31838	44626	68462	68462	1.94 Go
Juil 2003	0	0	0	0	0
Août 2003	0	0	0	0	0
Sep 2003	0	0	0	0	0

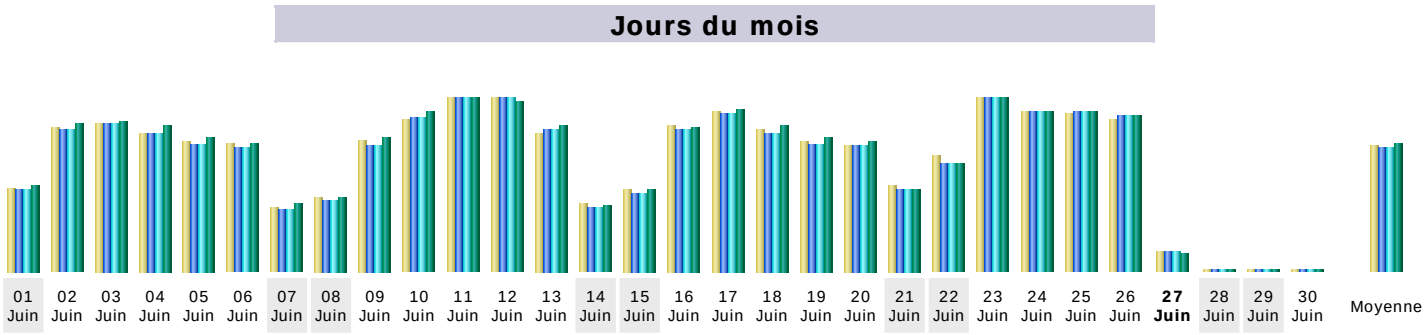
Oct 2003	0	0	0	0	0
Nov 2003	0	0	0	0	0
Déc 2003	0	0	0	0	0
Total	173050	241112	372212	372212	10.80 Go

TAB 2.3. ***RESUME DE STATISTIQUES APPLIQUEES SUR LE FICHIER LOG DU SITE : WWW.AWSTAT.NET***

Le tableau ci – dessus représente les différentes statistiques de navigation enregistrées dans le fichier Log du site www.awstats.net

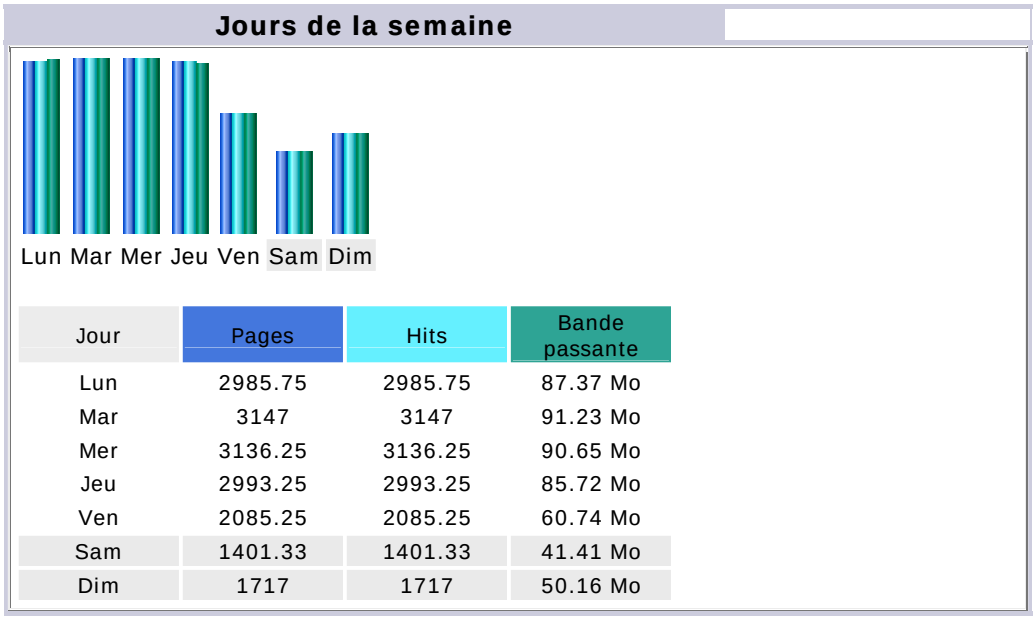
Tels que :

- Visiteurs différents : nombre de hotes (adresse IP) utilisés pour accéder au site (et voire au moins une page). Ce chiffre reflète le nombre de personnes physiques différentes ayant un jour accédées au site.
- Visites : on considère une nouvelle visite pour chaque arrivée d'un visiteur consultant une page et ne s'étant pas connecté pendant un moment.
- Pages : c'est le nombre de fois qu'une page du site est vue (cumul de tout visiteur, toute visite). Ce compteur et différent des hits car il ne comptabilise que les pages HTML et non les images ou autres fichiers.
- Hits : c'est le nombre de fois qu'une page, image, fichier du site est vu ou téléchargé par un visiteur. Ce compteur est donné à titre indicatif, le compteur page étant préféré.
- Bande passante : représente le nombre d'octets téléchargés lors des visites du site.



TAB 2.4. ***STATISTIQUE DU FICHIER EN DETAIL SUR UN MOIS DE L'ANNEE (MOIS DE JUIN)***

TAB 2.4 représente la distribution du nombre de (visiteurs, visites, pages, hits, bande passante) par rapport à 30 jours de navigations dans le site.



Jour	Pages	Hits	Bande passante
Lun	2985.75	2985.75	87.37 Mo
Mar	3147	3147	91.23 Mo
Mer	3136.25	3136.25	90.65 Mo
Jeu	2993.25	2993.25	85.72 Mo
Ven	2085.25	2085.25	60.74 Mo
Sam	1401.33	1401.33	41.41 Mo
Dim	1717	1717	50.16 Mo

TAB 2.5. ***STATISTIQUES APPLIQUEES EN MOYENNE SUR LA SEMAINE***

TAB 2.5. illustre des informations sur la navigation dans le site, sur tous les jours de la semaine. Ces chiffres représentent des moyennes.

Durée des visites		
Visites: 44626 - Moyenne: 252 s	**Visites**	**Pourcentage**
0s-30s	32280	72.3 %
30s-2mn	2436	5.4 %
2mn-5mn	2936	6.5 %
5mn-15mn	3215	7.2 %
15mn-30mn	1677	3.7 %
30mn-1h	1548	3.4 %
1h+	440	0.9 %
Inconnu	94	0.2 %

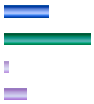
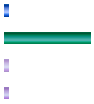
TAB 2.6. ***DIFFERENTS INTERVALLES MONTRANT LES DUREES DE VISITES DES INTERNAUTES SUR LE SITE.***

TAB 2.6. montre le pourcentage des visites consommées par les utilisateurs dans différents intervalles temporels.

Types de fichiers					
Types de fichiers			**Hits**	**Pourcentage**	**Bande passante**
	php	HTML dynamic page or Script file	62212	90.8 %	1.88 Go
	html	HTML static page	5976	8.7 %	47.96 Mo
?	Inconnu		135	0.1 %	4.15 Mo
-	net		126	0.1 %	5.35 Mo
	htm	HTML static page	7	0 %	475.88 Ko
-	mht		4	0 %	0
	cgi	HTML dynamic page or Script file	1	0 %	78.36 Ko
	pl	HTML dynamic page or Script file	1	0 %	0

TAB 2.7. ***DIFFERENTS TYPES DE FICHIERS ENREGISTRES DANS LE FICHIER LOG***

Dans TAB 2.7, nous montrons les types de fichiers d'Urls enregistrés dans le fichier Log.

es-URL (Top 10)					
42 pages différentes	Pages vues	Taille moyenne	Entrée	Sortie	
/	62195	31.76 Ko	44121	41569	
/awstats_contrib.html	5041	8.88 Ko	283	2406	
/awstats_whp.html	619	4.14 Ko	37	262	
/awstats_support.html	248	4.66 Ko	17	140	
http://awstats.sourceforge.net/	126	43.46 Ko	83	42	
/awstats_award.html	66	8.87 Ko	7	27	
/translate_c	42	45.74 Ko	17	24	
/search	25	33.74 Ko	16	13	
'	22		22	22	
/babelfish/urltrurl	12	54.62 Ko	1	3	
Autres	66	29.30 Ko	22	17	

TAB 2.8. ***LES 10 PREMIERES PAGES LES PLUS VISITEES DANS LE FICHIER LOG. .***

Enfin, les 10 première pages qui sont visitées dans le site et leurs apparitions comme page d'entrée ou sortie, sont illustrées dans TAB 2.8.

2.5 Pré-traitement

Du fait de son important volume d'informations, le fichier Log nécessite une épuration sur toutes les entités, qui ne sont pas nécessaires pour le traitement suivi d'un codage numérique les synthétisant.

Avant de concevoir notre corpus d'apprentissage nous commençons par nettoyer le fichier Log en éliminant les informations suivantes :

-
- Les lignes ayant des méthodes autres que GET, HEAD.
- Les enregistrements des fichiers images (*.gif, *.jpg, etc) qui sont typiquement présents dans d'autres pages et sont transmises à l'utilisateur à chaque demande d'une page.
- Les lignes contenant des erreurs de transmission (hors 200).
- Les champs différents des informations sur : le temps, l'adresse IP et l'URL.
- Les enregistrements ayant des URLs moins fréquentées.

Remarque :

Le temps que l'utilisateur met en consultant une page pourrait être un bon indicateur sur le comportement des utilisateurs. Cependant, le temps extrait à partir du fichier Log contient le temps de transfert de la requête dans le serveur. Donc, il serait « biaisant » de le prendre en compte pour l'analyse comportementale.

2.6 La notion de session

Notre tableau de données regroupe plusieurs sessions de plusieurs utilisateurs d'un site. Une session d'utilisateur est définie comme une séquence temporelle d'accès aux pages du site par un seul utilisateur. L'identificateur de cet utilisateur est rarement fourni par le serveur. Pour cette raison, nous définissons une session utilisateur comme un accès de la même adresse IP, pourvu que le temps entre deux pages consécutives ne dépasse un seuil de visualisation.

Chaque URL dans le site est assignée par un index $j \in \{1, \ldots\ldots, N\}$

Où ***N*** représente le nombre total de pages.

Globalement, une session peut être codée de la manière binaire suivante :

$$S_j^{(i)} = \begin{cases} 1 & \textit{si l'utilisateur demande la } j^{ème} \textit{ classe d'URLs durant la } i^{ème} \textit{ session} \\ 0 & \textit{sinon} \end{cases}$$

Cependant, ce codage ne traite pas les mouvements dans le site et n'explicite pas l'intérêt de l'utilisateur devant les pages.

2.7 Codage

Le fichier Log présenté dans sont état brut, contient un volume de données important et semi – structuré. Pour cette raison, nous estimons qu'une méthode de codage soit nécessaire pour attribuer un format spécifique aux données.

2.7.1 Le codage relationnel

Ce codage est utilisé pour les sites à structure arborescente. Il consiste à regrouper les pages similaires d'un site en utilisant l'agrégation des mesures de similarités sur les variables liées à la page. Deux variables caractérisent chaque page d'un site : l'adresse URL et le contenu.

2.7.1.1 L'adresse URL

On modélise le site Web sous forme d'un arbre dont les nœuds représentent les différentes URLs.

Les nœuds sont essentiellement la structure des répertoires organisées dans le serveur du site avec des liens explicitement introduits dedans. La branche connecte un nœud à un autre si l'URL correspondant à ce dernier est hiérarchiquement localisé au dessus du premier (exemple /Account et /Account/Corporate). La racine de l'arbre correspond à la syntaxe du nom du site (http://).

Se focalisant sur la représentation syntaxique de deux URLs, leur similarité est estimée par la comparaison de leurs localisations dans l'arbre. Cette comparaison peut être déterminée par le chemin de l'arbre à partir de la racine jusqu'à la feuille.

Chaque répertoire possède une sémantique et contient des pages qui sont reliées suivant le point de vue du concepteur. De plus, les

répertoires les plus profonds contiennent des informations plus précises que les répertoires de surface. Nous utilisons alors la fonction élémentaire de similarité suivante qui prend en compte ces propriétés:

$$S_{Adresse}(x, y) = 1 - \frac{f(x,g(x,y)) + f(y,g(x,y))}{f(x,racine) + f(y,racine)}$$

Dans cette fonction, une adresse est considérée comme la référence d'un nœud d'une hiérarchie de symboles. La fonction ***g*** donne le nœud commun le plus spécifique des deux adresses et la fonction ***f*** donne le nombre de liens entre deux nœuds.

Exemple

http:// → www.wstore/ → Account/ → LoginOpen.asp. .. ..(1)
Account/ → Corporate/ → profil.asp. ..(2)

$$S_{Adresse}(1, 2) = 1 - \frac{1+2}{3+4} = 0.57$$

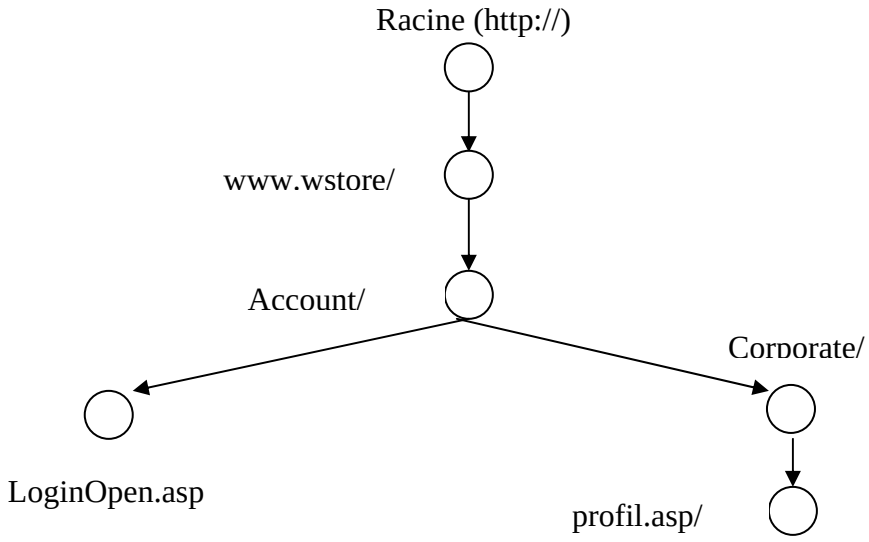


FIG 2.3. *MODELISATION D'UNE PARTIE DU SITE EN ARBRE.*

2.7.1.2 Le contenu de la page

Le contenu d'une page Web est décrit par du texte brut augmenté de directives de formatage, données en langage HTML. Ces directives sont interprétées par le navigateur de manière à afficher le texte avec différents effets : hyperliens, insertion d'images, changement de la taille de la fonte, titre, sous-titre, liste numérotée…etc. Plus

précisément, le texte est enrichi de descripteurs (tags) qui apportent principalement des informations de mise en page et de structuration. Par exemple, sur la figure FIG 2.4, les deux descripteurs <B> et </B> permettent de définir une zone de texte à afficher en gras (Bold). Sur le plan de la structuration, les descripteurs <H1> </H1>, <H2> </H2>, <H3> </H3> permettent de définir des titres de niveaux respectifs 1, 2 et 3. De plus, une page est divisée en deux parties principales : une partie descriptive identifiée par les descripteurs <HEAD> et </HEAD> et une partie informative entre les descripteurs <BODY> et </BODY>. La partie descriptive donne le titre de la page entre les descripteurs <TITLE> et </TITLE> ainsi que des métadonnées (MetaData) sur les pages dans des descripteurs <META…> : mots clés, description de la page, nom de l'auteur, Logiciel utilisé pour écrire cette page. La partie informative contient le texte formaté à afficher. Ces deux parties sont réunies dans une zone encadrée par les descripteurs <HTML> et </HTML>. Et la page commence par un descripteur <!DOCTYPE… > donnant la version du format utilisé.

```
<HTML>
<HEAD>
<TITLE>.....
....
<META NAME="keywords"
        CONTENT= "WSTORE, Compaq,
                  Mico-Informatique">
<META NAME= "description"
        CONTENT= "WSTORE"
</HEAD>
<BODY>
...
<H1> ..... </H1>
...
<H1> ..... </H1>
...
</BODY>
</HTML>
```

FIG 2.4. ***EXEMPLE DE STRUCTURATION D'UNE PAGE HTML.***

On peut remarquer qu'une page Web donne un ensemble de mots clés grâce au descripteur <META>. A vrai dire, ce descripteur n'est pas suffisant pour définir tous les mots clés dans le domaine du traitement de l'information. Des travaux en cours étudient la normalisation de traitements automatiques des métadonnées en vue de meilleures possibilités de traitements automatiques des pages Web. Dans notre cas, loin de ce cadre de travail, nous nous

restreignons aux informations extraites du descripteur décrit ci-dessus avec l'aide du concepteur du site.

On défini donc, la similarité entre le contenu de deux pages *x* et *y* en prenant en compte le nombre de mots clés communs entre elles :

$$S_{Contenu}(x, y) = \begin{cases} 0 & si\ x \cap y = \phi \\ \dfrac{Card(x \cap y)}{card(x \cup y)} & sinon \end{cases}$$

Ayant définit ces deux fonctions de similarité, nous les pondérons avec un vecteur de poids ($w_{Adresse}$, $w_{Contenu}$) pour calculer enfin la similarité entre deux pages dans le site.

$$S_{(Adresse,\ Contenu)}(x,y) = w_{Adresse}\, S_{Adresse} + w_{Contenu}\, S_{Contenu}$$

On considère que le poids sur le contenu est plus important que celui sur l'adresse. En effet, le contenu d'une page représente plus d'information que la hiérarchie de l'adresse.

Par conséquent, nous pouvons définir une matrice de similarité entre les pages.

2.7.2 Une nouvelle technique de codage : Le codage par matrice quasi-comportementale

Le codage décrit précédemment n'est pas adéquat à la plupart des sites conçus aujourd'hui à cause de leurs structures aléatoires dues au dynamisme. Pour cela, nous avons développé une autre méthode de codage à partir du fichier Log, qui consiste à caractériser une page par son importance de passage. En d'autres termes, par son poids de précédence et de succession par rapport aux autres pages du site apparues dans le fichier Log.

Le principe de cette méthode consiste à calculer pour chaque page sa fréquence de précédence et de succession par rapport à toutes les autres pages (TAB 2.9).

P	**Entrée**	**P_1……. .. .P_j ……. ….P_N**	**S**	**P_1.....P_k…….…P_N**	**Sortie**
Entree	MaxE	0 0 0 ……. ……..0		e_1..... ...e_k.....e_n	0
P_1 ⋮ P_i ⋮ P_N	a	..... b		..... c	d
Sortie	0	s_1.....s_k.....s_n		0 0 0 …. …….0	MaxS
		Précédence		Succession	

TAB 2.9. ***CODAGE PAR MATRICE QUASI-COMPORTEMENTALE***

MaxE : le nombre total de pages apparues comme entrée.

MaxS : le nombre total de page apparues comme sortie.

P_i P P_j : P_i précède P_j

P_i S P_j : P_i succède P_j

Exemple :

La page p_i est apparue *a* fois comme page d'entrée, précédée *b* fois par P_j, succédée *c* fois par P_k et apparue *d* comme page de sortie.

2.7.3 Incorporation de la dynamique dans la matrice quasi-comportementale

On rappelle que 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 consistait donc à calculer la matrice de précédence et de succession sur la totalité du fichier Log. Grâce à ce codage, on a pu réussir à caractériser chaque page du site selon sa présence dans les navigations enregistrées dans le fichier. Ce codage tel qu'il a été décrit cause un problème d'effectif. En effet, n'ayant pas 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, nous avons proposé de glisser la matrice sur le mois. En d'autres termes, nous avons calculé des matrices quasi-comportementales par jour, voire même par semaine ou généralement par pourcentage de partitionnement sur la base. Cette méthode nous permet de multiplier le nombre d'échantillons et d'avoir plusieurs exemples pour chaque URL (TAB 2.10).

			Précède	Suit	
Individus(URLs)	jour	E	URL_1 URL_2 ...URL_j . .. . URL_N	URL_1 URL_2 .... URL_l . ..URL_N	S
E_1 URL_1^1 URL_2^1 ⋮ URL_i^1 ⋮ URL_N^1 S_1 E_2 URL_1^2 URL_2^2 ⋮ URL_i^2 ⋮ URL_N^2 S_2 ⋮ E_i URL_1^i URL_2^i ⋮					
URL_i^i	*k*	*a*	..... *b*	..... *c*	*d*
⋮ URL_N^i S_i ⋮					

TAB 2.10. ***TABLEAU DE CODAGE EN UTILISANT LA MATRICE QUASI-COMPORTEMENTALE DYNAMIQUE***

Dans le tableau ci-dessus, on montre un exemple de codage d'une URL (URL_i) comme suit:

Dans le $k^{iéme}$ jour du fichier l'URL_i^i est apparue *a* fois comme page d'entée, précédée *b* fois par la page URL_j, suivie *c* fois par la page URL_l et apparue *d* fois comme page de sortie.

Exemple :

Étant donné un fichier Log générant 40 URLs sur 30 jours.

Si on glisse la matrice sur le mois jour par jour, on obtient (40+2) fois 30 soit une base de 1260 exemples.

Non seulement, on caractérise les pages par des variables comportementales i.e. tel que les internautes perçoivent le site. Mais on multiplie aussi les informations sur les pages en codant leurs contenus comportementaux en détails jour par jour.

2.8 Application : analyse quasi – comportementale des sessions de navigations

Pour notre application, nous utilisons un site commercial nommé : www.123credit.com. Il s'agit d'un site qui commercialise un certain nombre de services pour des clients désirant avoir des crédits. En premier temps, nous recevons le fichier Log du site dans son état brut (FIG 2.5). Ce fichier décrit des transactions de navigation pendant 1 mois. Son volume voisine les 700 Méga octets

```
195.154.37.61 - - [30/Jun/2000:23:59:07 +0200] "GET /123credit/gene/image/mpasson.gif HTTP/1.1" 200 189
212.155.170.16 - - [30/Jun/2000:23:59:07 +0200] "GET /123credit/gene/image/retoff.gif HTTP/1.1" 404 359
212.155.170.16 - - [30/Jun/2000:23:59:02 +0200] "POST /cgi-bin/bvsm/123credit/sous/scripts/register_c.jsp HTTP/1.1" 200 -
195.154.37.61 - - [30/Jun/2000:23:59:07 +0200] "GET /123credit/gene/image/mpassoff.gif HTTP/1.1" 200 276
195.154.37.61 - - [30/Jun/2000:23:59:11 +0200] "GET /123credit/gene/image/suiton.gif HTTP/1.1" 404 358
212.155.170.16 - - [30/Jun/2000:23:59:06 +0200] "GET /123credit/gene/image/suiton.gif HTTP/1.1" 404 359
195.154.37.61 - - [30/Jun/2000:23:59:04 +0200] "GET /123credit/gene/javascript/validation.js HTTP/1.1" 200 1404
195.154.37.61 - - [30/Jun/2000:23:59:02 +0200] "POST /cgi-bin/bvsm/123credit/accu/scripts/accu_gene.jsp HTTP/1.1" 200 -
195.154.37.61 - - [30/Jun/2000:23:59:12 +0200] "GET /123credit/gene/image/suitoff.gif HTTP/1.1" 404 358
212.155.170.16 - - [30/Jun/2000:23:59:05 +0200] "GET /123credit/gene/image/croixon.gif HTTP/1.1" 404 359
193.252.35.200 - - [30/Jun/2000:23:59:13 +0200] "GET /123credit/gene/image/mpasson.gif HTTP/1.1" 200 189
212.155.170.16 - - [30/Jun/2000:23:59:06 +0200] "GET /123credit/gene/image/suitoff.gif HTTP/1.1" 404 359
212.155.170.16 - - [30/Jun/2000:23:59:07 +0200] "GET /123credit/gene/image/reton.gif HTTP/1.1" 404 359
193.252.35.200 - - [30/Jun/2000:23:59:12 +0200] "GET /123credit/gene/image/okoff.gif HTTP/1.1" 200 296
195.154.37.61 - - [30/Jun/2000:23:59:06 +0200] "GET /123credit/gene/image/okoff.gif HTTP/1.1" 200 296
195.154.37.61 - - [30/Jun/2000:23:59:06 +0200] "GET /123credit/gene/image/okon.gif HTTP/1.1" 200 259
195.154.37.61 - - [30/Jun/2000:23:59:08 +0200] "GET /123credit/gene/image/lecton.gif HTTP/1.1" 200 126
193.252.35.200 - - [30/Jun/2000:23:59:13 +0200] "GET /123credit/gene/image/mpassoff.gif HTTP/1.1" 200 276
195.154.37.61 - - [30/Jun/2000:23:59:09 +0200] "GET /123credit/gene/image/croixon.gif HTTP/1.1" 404 358
193.252.35.200 - - [30/Jun/2000:23:59:11 +0200] "GET /123credit/gene/javascript/validation.js HTTP/1.1" 200 1404
212.155.170.16 - - [30/Jun/2000:23:59:05 +0200] "GET /123credit/gene/image/croixoff.gif HTTP/1.1" 404 359
193.252.35.200 - - [30/Jun/2000:23:59:12 +0200] "GET /123credit/gene/javascript/rollover.js HTTP/1.1" 200 1614
195.154.37.61 - - [30/Jun/2000:23:59:05 +0200] "GET /123credit/gene/javascript/rollover.js HTTP/1.1" 200 1614
……
……..
195.154.37.61  - -    [29/Jui/2000:23:59:12 +0200]    "GET /123credit/gene/image/reton.gif HTTP/1.1"  404   358
……..
```

FIG 2.5. ***UN MORCEAU DU FICHIER LOG DU SITE*** **123*CREDIT.COM***

Nous pouvons remarquer la nature semi-structurée du fichier. Certes, les enregistrements sont organisés sous forme de plusieurs attributs, mais peu d'entre eux sont nécessaire pour l'analyse. D'où l'utilisation d'une procédure d'épuration selon le principe décrit dans la section 2.4.

Nous appliquons par la suite, la procédure du sessionage avec un seuil de visualisation de 30 secondes. Nous extrayons donc, un nombre de sessions égal à 37174 dont le nombre d'Urls est égal à 40. Le fichier commence donc à perdre de sa taille. Une session est une succession d'enregistrements ayant la même adresse IP et ne dépassant pas le seuil de visualisation.

Après avoir codifié les Sessions de 1 à 37174 et les Urls de 1 à 40 plus les deux Urls Entrée (E) et Sortie (S) nous générons le tableau suivant :

Identificateur de session	Jour	Urls de session
1	[30/Jun/2000]	E 24 2 4 0 12 1 …… S
2	[30/Jun/2000]	E 1 4 8 12 10 4 17 26 ….S
3	[30/Jun/2000]	……
…. .	. .. .	……. .
	[01/Jui/2000]	
	. .. .	. .. .
37174	[29/Jui/2000]	E 7 14 7 2 8 3 . . S

TAB 2.11. ***TABLEAU DES SESSIONS ISSUES DU FICHIER LOG***

Nous pouvons remarquer dans le tableau ci-dessus, que les transactions dans le fichier Log sont enregistrées dans une période de 30 jours : du 30 juin 2000 au 29 Juillet 2000.

Selon le principe décrit dans 2.6.3, nous pouvons avoir un tableau de données de ((40 + 2) x 30) exemples, caractérisés par ((40×2)+2) Urls.

Soit donc, un tableau de 1260 individus fois 82 variables.

Il est bien évident que ce dernier tableau contient des entiers discrets. Nous normalisons le tableau en colonnes pour centrer et réduire les données. Nous proposons aussi, de normaliser le tableau en lignes pour éliminer toute dépendance entre les variables.

Par conséquent, le tableau de données (ayant la forme décrite dans TAB 2.11) sera d'une taille de quelques Kilos octets.

Etant finalement prêtes, les données seront traitées par la suite par des techniques de classification automatique. Ces dernières sont de type neuronale et servent de modèles de regroupement, de codage et de visualisation.

2.9 Conclusion

Dans toute analyse de type Data – Mining, un expert doit recueillir des grands volumes de données principalement issues de fonctions de production, et stockées sous forme de bases de données. Sur Internet, c'est l'utilisateur qui va fournir ces éléments par sa simple navigation.

Pour des intérêts techniques ultérieures les données doivent être nettoyées. D'où l'utilisation d'un processus de pré – traitement dont résulte le fichier dit épuré.

Finalement, un processus de formatage a été dévellopé pour préparer les données afin d'en optimiser l'exploitation, selon la nature de l'analyse souhaitée et de la technique de Data Mining à mettre en œuvre.

Chapitre III

3 Classification non supervisée : Visualisation

Nous décrivons dans ce chapitre deux méthodes de classification automatique. La première est dite neuronale et consiste à utiliser l'algorithme de Kohonen. La seconde est dite hiérarchique et consiste à utiliser un arbre de classification respectant un critère de métrique. Les deux méthodes sont utilisées de manière complémentaire, pour définir un outil de visualisation compréhensive des données comportementales.

3.1 Introduction

Parmi les phénomènes collectifs émergeant naturellement de la propagation d'activation, la compétition tient une place importante et utile. Compte tenu de l'intérêt de ce type de phénomène, il n'est pas surprenant de constater que la compétition est beaucoup étudiée en connexionisme, et ce, dès les premiers travaux. En effet, certains des travaux de Rosenblatt **[48]** faisaient explicitement usage de la compétition connexionniste. Les caractéristiques de la compétition connexionniste furent étudiées de façon théorique dès la fin des années 60, notamment par Grossberg et ses collègues **[34]**.

3.2 Apprentissage supervisé et non supervisé

On distingue en général deux familles d'algorithmes neuronaux : des algorithmes supervisés et des algorithmes non supervisés. Le concept d'apprentissage supervisé repose sur l'existence de couples d'apprentissage (z, y), où y est la réponse attendue du réseau lorsqu'on lui présente l'entrée z. Un exemple typique d'application à apprentissage supervisé, est le classement où y est le numéro de la classe à laquelle appartient z.

Il existe également de nombreux problèmes dans lesquels on dispose d'un ensemble de données $A=\{z_1, z_2, \ldots, z_N\}$ sans qu'aucune « étiquette » soit attachée à z. On souhaite cependant extraire de l'information pertinente d'un tel ensemble, par exemple :

- Un groupement des exemples en « classes » tels que les exemples se retrouvent dans une même classe se ressemblent fortement. C'est le problème de classification automatique.
- Une réduction de dimensionalité de l'espace mettant en évidence de façon visuelle les regroupements significatifs des exemples. C'est l'objet de la quantification vectorielle.
- Une approximation de la densité de probabilité sous – jacente au processus de génération de données.

Comme on n'associe pas de réponse connue a priori à une forme d'entrée, il ne peut y avoir supervision de la part de l'utilisateur au cours de l'apprentissage. On dit que notre apprentissage est non supervisé **[29]**.

3.3 Un réseau à compétition simple

3.3.1 Définition d'un neurone formel

Un neurone formel se présente comme un opérateur effectuant des opérations élémentaires simples. Il est muni de *n* entrées et *n* paramètres qui représentent les poids des connections entre le vecteur d'entrée et le neurone *j* (FIG 3.1).

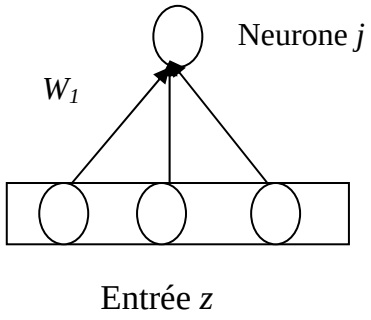


FIG 3.1. ***NEURONE FORMEL (D'APRES [29], P12).***

3.3.2 Activation d'un neurone formel

Dans la plupart des modèles neuronaux, le produit scalaire entre vecteur poids *W* et vecteur d'entrée *z* est la formule la plus classique pour calculer le niveau d'activation d'un neurone : *W* et *z* sont similaires si leur produit scalaire *Wz* est grand. Une modification par rapport à cette démarche classique permet d'utiliser la distance euclidienne. Au lieu de considérer le maximum d'activité (maximum du produit scalaire), on recherche le minimum des distances $d(W,z)$ où d est une distance.

Si les poids sont normalisés, la valeur pour laquelle le produit scalaire est maximum est aussi celle qui est la plus proche de *z* au sens de la distance euclidienne.

3.3.3 Architecture compétitive

Soit le réseau suivant, composé d'une couche de neurones qui reflète passivement les patrons d'entrées présentés au réseau, et une couche de neurones de sortie en compétition :

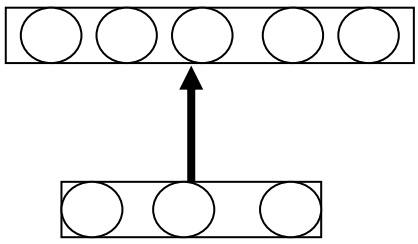


FIG 3.2. UN RESEAU A COMPETITION SIMPLE *(D'APRES [34], P112)*.

Peu de conditions sont nécessaires pour que des neurones fassent preuve de compétition. Il suffit de :

- S'assurer que ceux – ci répondent de façon différente aux diverses entrées qui leur sont présentées. En général, l'aléa initial des poids synaptiques initiaux assure que cette contrainte soit respectée.
- Que les fonctions d'activation des neurones soient bornées.
- D'un mécanisme de concurrence entre neurones. Habituellement, des connexions (inhibitrices) entre les neurones servent à cet effet.

On utilise le réseau de la façon suivante : un patron d'entrée est présenté au réseau, provoquant des réponses variées de la part des neurones de sortie. La compétition s'installe alors entre ces derniers, et prend la forme d'un combat d'influence qui doit éventuellement se stabiliser grâce à la force des liens inhibiteurs. A la fin de la compétition, le ou les neurones de sortie les plus activés sont déclarés « vainqueurs ».

3.3.4 Apprentissage compétitif

Le principe de l'apprentissage compétitif est *d'encourager le vainqueur* : les poids d'entrée du neurone gagnant la compétition, sont rapprochés du vecteur d'entrée responsable de sa victoire. Ainsi, l'apprentissage améliorera les chances de victoire du neurone lors de la prochaine présentation du même vecteur. Cependant, cette même modification risque de diminuer ses chances pour d'autres entrées.

Parce qu'il apprend à gagner pour un petit nombre d'entrées semblables, le neurone agit à terme comme un *détecteur* de ce type d'entrées. On voit alors l'utilité de l'apprentissage compétitif qui permet au réseau de développer – sans autres informations que les données elles-mêmes – des capacités de *classification* ou de *détection de traits*.

3.4 Le modèle des cartes Auto – Organisées de Kohonen

3.4.1 Structure topologique

La carte topologique de Kohonen « Self – Organizing Map ou SOM » impose une structuration supplémentaire sur ses neurones. Cette structuration lie les neurones ensemble et les contraint à respecter une certaine topologie lors de l'apprentissage. Il en résulte que le comportement final des neurones varie graduellement d'une région de la carte à l'autre, plutôt qu'arbitrairement comme c'est en général le cas dans les réseaux compétitifs.

Dans la carte topologique de Kohonen, les neurones sont répartis aux nœuds d'un maillage, par exemple à mailles carrés ou hexagonales, et constituent le réseau. Les nœuds du réseau sont indexés par des nombres entiers, qui définissent une relation de voisinage entre neurones.

La structure topologique peut avoir plusieurs formes. Dans le cas pratique la structure la plus utilisée est sous forme d'une grille carrée. D'autres formes de topologie peuvent être choisies (hexagonale). Nous présentons sur la FIG 3.3 un maillage carré en dimension 2, un neurone *c*, ses voisins les plus proches d'ordre 1 et ses voisins d'ordre 2.

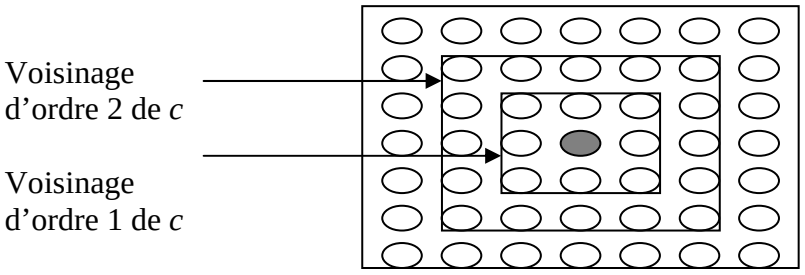


FIG 3.3. ***TOPOLOGIE A MAILLAGE CARRE (D'APRES [29], P13)***

La structure topologique et son système de voisinage permettront la coopération d'une cellule avec ses voisins, et donc la mise en œuvre du principe de localisation et de compétition.

Plus formellement on définit la topologie de la carte à l'aide d'un graphe non orienté $G(C,U)$ pour lequel l'ensemble des sommets C est constitué par l'ensemble des neurones et l'ensemble U contient toutes les arêtes reliant des voisins directs. Cette topologie permet de

définir une distance sur le graphe. En effet, on définit la distance $\delta(i,j)$ entre deux cellules *i* et *j* de la carte comme étant la longueur du chemin le plus court qui sépare la cellule *i* à la cellule *j* (FIG 3.4)

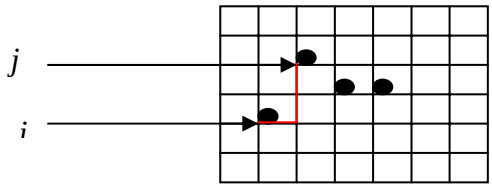


FIG 3.4. ***DISTANCE δ DEFINIE SUR LA CARTE δ (J, I)=3 (D'APRES [29], P13).***

La distance $\delta(i, j)$ est appelée distance « échiquier »

Pour reproduire la fonction « chapeau mexicain » (FIG 3.5) qui montre que l'influence d'un neurone *i* sur un neurone *j* dépend de leur proximité, on utilise une fonction *K*(.) définie par :

$$K : CxC \longrightarrow R \longrightarrow R^{+}$$
$$(i,j) \longrightarrow \delta(i,j) \longrightarrow K(\delta(i,j))$$

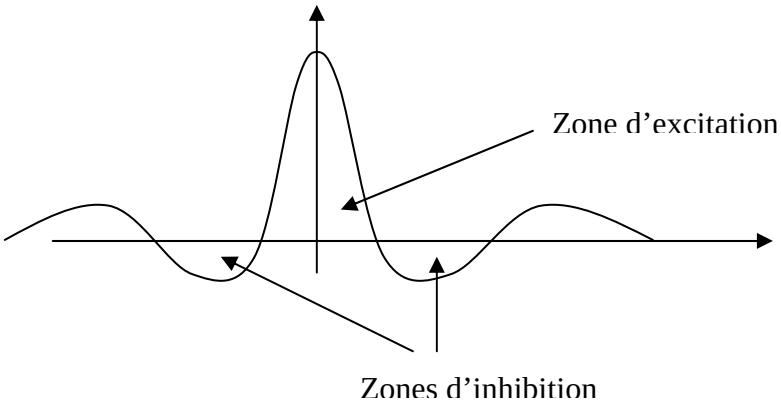


FIG 3.5. ***CHAPEAU MEXICAIN : POINTS DES INTERACTIONS ENTRE LE NEURONE CENTRALE I ET SES VOISINS (D'APRES [29], P10).***

La première version de l'algorithme de Kohonen utilise la fonction indicatrice qui définit un voisinage en « tout ou rien » (FIG 3.6)

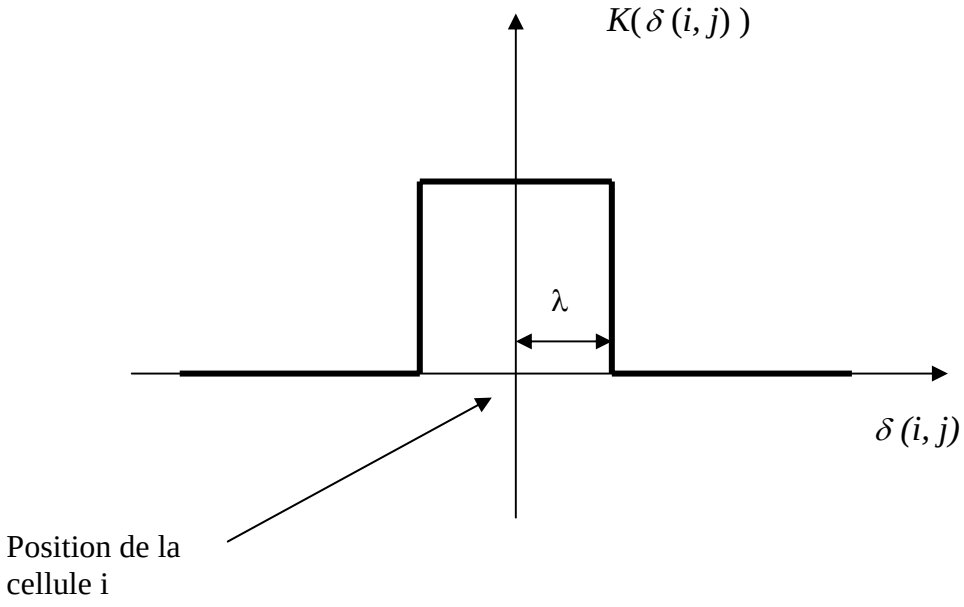


FIG 3.6. FONCTION VOISINAGE A *SEUIL (D'APRES [29], P14)* .

Ce voisinage s'écrit en utilisant la distance « échiquier » $\delta(i, j)$:

$$\begin{cases} K(\delta(i,j)) = 1 \quad si \quad \delta(i,j) < \lambda \\ 0 \quad sinon \qquad\qquad où\ \lambda\ est\ un\ entier \end{cases}$$

La portion de la grille où $K(.) = 1$ est ainsi un carré de largeur 2λ centré sur la cellule i. L'influence sur i des cellules se trouvant au delà de ce carré est nulle. On pourrait faire évoluer ce voisinage au cours du temps en considérant λ comme fonction décroissante en fonction du temps t. Si $\lambda(t)$ décroît quand t croît, le voisinage aura la même forme avec de moins en moins de cellules qui influenceront la cellule i au cours du temps.

3.4.2 Architecture

La carte topologique est composée essentiellement d'une couche compétitive de neurones de sortie. Ces neurones sont alimentés par une couche de neurones d'entrée qui font que passivement refléter le patron d'entrée présenté au réseau.

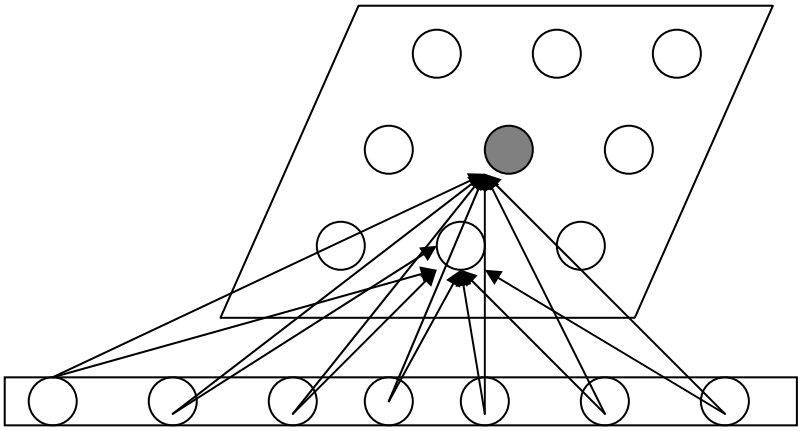


FIG 3.7. ***ARCHITECTURE A DEUX COUCHES : REPRESENTATION D'UNE CARTE TOPOLOGIQUE DE DIMENSION 2 (D'APRES*** *[29]***, P15).**

Cette structuration n'est pas explicitement présente dans l'architecture du réseau lors de sa conception. En effet, tout l'intérêt de ce type de réseau est que la notion de voisinage se développe progressivement par apprentissage et n'est apparente que dans le comportement des neurones du réseau.

3.4.3 Une heuristique pour déterminer la topologie dans SOM

Le choix du nombre de neurones initial reste toujours un souci pour toute méthode de classification. Pour les cartes topologiques, Kohonen propose une heuristique qui calcule le nombre de neurones initial ainsi que les dimensions de la carte. Le nombre de neurones est estimé comme suit :

$$C = 5 \times N^{0,54321}$$

Où N est le nombre d'exemples dans la base.

Pour les dimensions de la carte (x, y), nous résolvons le système d'équation suivant :

$$\begin{cases} x \times y = N \\ x / y = a / b \end{cases}$$

Où a et b représentent les plus grandes valeurs propres de la matrice de covariance de la base d'apprentissage.

On peut remarquer que C est calculé en fonction du nombre d'exemples d'apprentissage mais ne considère en aucun cas la nature

de cette base. Autrement dit, pour une grande base contenant des exemples redondants, la valeur de *C* serait importante ce qui créerait des neurones « supplémentaires » dans la carte.

Pour remédier à ce problème nous avons proposé la mesure heuristique suivante :

Connaissant la topologie de la carte, rectangulaire soit elle ou bien hexagonale, les deux neurones les plus éloignés l'un de l'autre, sont le neurone bas – gauche et le neurone haut –droit de la carte. A partir de ce principe, nous considérons que les deux formes les plus éloignées dans l'espace réel des données devraient être respectivement projetées sur ces deux neurones.

Soient donc les deux formes suivantes : x_i, x_j prises d'un ensemble d'apprentissage à N formes telles que :

$$\|x_i\| \leq \|x_k\|, \ \forall 1 \leq k \leq N.$$

$$\|x_i - x_j\| \geq \|x_i - x_l\|, \ \forall 1 \leq l \leq N.$$

On peut remarquer que x_i représente la plus proche valeur par rapport à l'origine et x_j représente le point le plus éloigné par rapport à x_i (FIG 3.8).

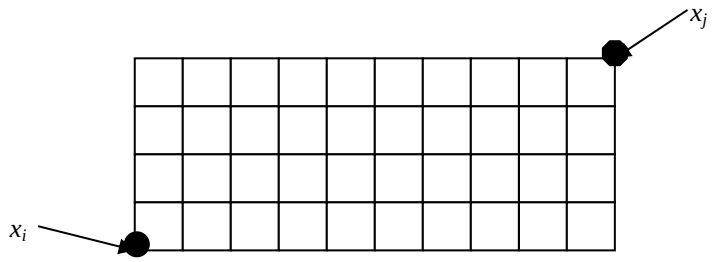


FIG 3.8. ***POSITIONNEMENT DU POINT LE PLUS ELOIGNE PAR RAPPORT AU POINT LE PLUS PROCHE A L'ORIGINE***

Nous exprimons notre heuristique comme suit :

Soit la forme x_m telle que :

x_m est la 3ème plus proche forme par rapport à xi (x_m est supposée être sur la diagonale). Nous proposons donc, l'heuristique suivante :

$$C = \left(\frac{\|x_i - x_j\|}{\|x_i - x_m\|}\right)^2.$$

Cette heuristique fournit un nombre de neurones qui tient compte de la nature des formes et pas de la taille de la base. Nous l'appelons : HSOM

a et *b* sont calculés de la même manière que précédemment.

3.4.4 Apprentissage

Intuitivement, l'apprentissage se poursuit de la façon suivante : quand un neurone gagne la compétition, ses poids sont modifiés selon une règle compétitive, et il se spécialise progressivement dans la détection d'un type de données. Cependant, les neurones autour du vainqueur subissent aussi un apprentissage, et sont donc entraînés au même type de réponse. Cela a tendance à former une région, une « bulle » centrée autour du vainqueur, dont les neurones sont sensibles au même type de données. Comme l'apprentissage se poursuit, les neurones gagnent chacun leur tour, ce qui leur donne à tous une certaine spécificité. Cependant, parce que les neurones profitent aussi (à un moindre degré) de la victoire de leurs voisins, leur spécialisation n'est pas arbitraire : elle est influencée par celle des neurones environnants. C'est ce phénomène qui est responsable de la gradation des spécialisations observables le long de la carte.

Formellement, l'algorithme utilise une distance qui est la distance euclidienne, elle est utilisée pour attribuer une forme *z* à un neurone représenté par son vecteur de poids.

Chaque étape de l'algorithme est constituée de deux phases : une phase de compétition au cours de laquelle une forme *z* est affectée au neurone de la carte de plus proche au sens de la distance euclidienne et une phase d'apprentissage qui consiste à mettre à jour les poids dans un voisinage autour du neurone choisi par l'étape de compétition.

L'évolution temporelle de l'algorithme est indicée par l'entier *t*. L'algorithme est stochastique : on adapte les poids après chaque présentation d'un exemple. L'influence du voisinage varie avec le temps par l'intermédiaire d'une fonction $\lambda(t)$ décroissante avec le temps.

Au temps *t*, on peut décrire les deux étapes de l'algorithme d'apprentissage par :

- **Etape1 :** Compétition entre les cellules de la carte : sélectionner le neurone de la carte le plus proche de *z* à partir d'une comparaison entre le vecteur d'entrée *z* et chaque vecteur W_j^t. le vecteur poids gagnant W_c^t est celui qui présente la plus

grande ressemblance à l'entrée z. Il est déterminé au sens de la distance euclidienne par :

$$\left\|z-W_c^t\right\|^2=\min_{j\in C}\left\|z-w_j^t\right\|^2$$

où la minimisation est prise sur l'ensemble des vecteurs de poids associés à chaque neurone du réseau.

- **Etape 2 :** Adaptation en rapprochant le poids de la cellule gagnante et de ses voisins de la forme z. Modifier les poids des neurones *j* selon le procédé décrit par la règle d'adaptation suivante :

$$W_j^t=W_j^{t-1} + \varepsilon(t)(z-W_j^{t-1}) \; si \quad \delta(c,j) < \lambda(t)$$
$$W_j^t= W_j^{t-1} \; sinon$$

où $\varepsilon(t)$ est le pas d'apprentissage. Il est défini comme fonction décroissante du temps *t*. Au cours du temps, le pas de correction des poids diminue.

Nous présentons donc, l'algorithme de Kohonen comme suit :

1. **Initialisation :** t = 0, initialiser les poids W_j, à des valeurs tirées aléatoirement

   au temps t , présenter la forme z sur les i caractéristiques et faire :

2. **Compétition :** choix du gagnant W_c^t :

$$\left\|z-W_c^t\right\|^2=\min_{j\in C}\left\|z-w_j^t\right\|^2$$

3. **Adaptation :** mise à jour des poids dans le voisinage du gagnant :

$$W_j^t=W_j^{t-1} + \varepsilon(t)(z-W_j^{t-1}) \; si \quad \delta(c,j) < \lambda(t)$$
$$W_j^t= W_j^{t-1} \; sinon$$

$t = t+1$, présenter une autre forme z et répéter les deux étapes de compétition et d'adaptation.

Une itération correspond à la présentation de toute la base d'apprentissage. Un critère d'arrêt peut être un nombre d'itération fixé a priori.

Lorsque la dimension de l'espace d'entrée est inférieure ou égale à 3, il est possible de représenter visuellement la position des vecteurs poids et les relations de voisinage direct entre deux cellules. Cette présentation permet de faire une appréciation visuelle de la carte. Elle fournit une information qualitative sur la qualité de la carte et le choix de son architecture. Mais, dans le cas des espaces de représentation de dimension, ce mode de représentation n'est plus utilisable.

En dimension deux, on définit deux axes gradués auxquels on associe les deux dimensions d'entrée et qui permettront de situer les poids des connexions en provenance des deux unités d'entrée dans l'espace des poids. On relie entre eux les points représentant des neurones voisins.

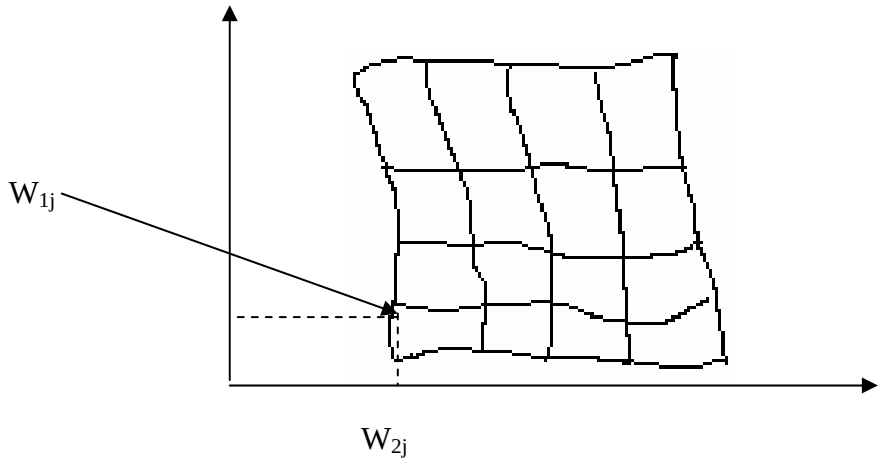


FIG 3.9. ***REPRESENTATION DANS L'ESPACE DES POIDS (D'APRES [29], P15).***

Exemple en dimension 2

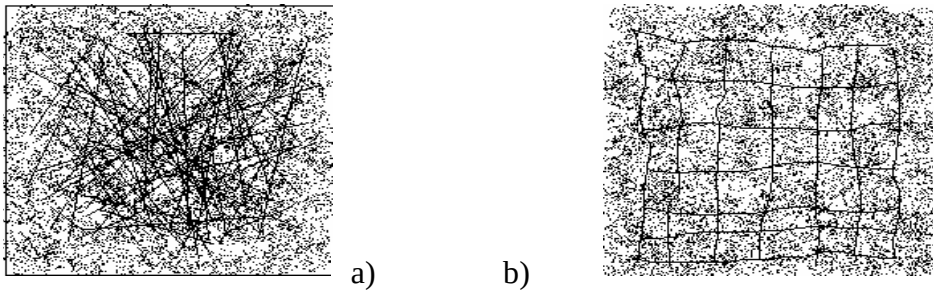


FIG 3.10. ***A) INITIALISATION ALEATOIRE DES POIDS B) STABILISATION DE LA CARTE APRES N ITERATIONS (D'APRES [29], P17).***

Certaines contraintes sont imposées pour assurer la convergence de l'algorithme. La première concerne l'évolution de la taille du voisinage au cours de l'apprentissage. Elle doit être assez grande au début, puis décroître au cours du temps. La seconde est la décroissance au cours du temps t du pas d'apprentissage $\varepsilon(t)$ pour assurer une stabilisation du processus.

3.5 Analyse du processus d'auto – organisation

L'algorithme de Kohonen possède de nombreuses propriétés qui ont été étudiées par Kohonen et par d'autres auteurs Ritter **[42]**, particulièrement sur la détermination des conditions nécessaires pour garantir la convergence du processus vers un état stable. L'analyse de cette stabilité fait apparaître l'influence de la distribution de probabilité des observations.

Il est possible dans certains cas de prouver qu'à l'état d'équilibre, les vecteurs poids des neurones approximent la distribution de probabilité des entrées tout en conservant un ordre par la topologie.

3.5.1 Convergence

Ritter et Shulten **[42]** ont dégagé deux conditions nécessaires sur la forme du pas d'apprentissage $\varepsilon(t)$ pour garantir la convergence de l'apprentissage vers un état stable. L'approche utilisée repose sur la modélisation de l'apprentissage par un processus Markovien, dont les états sont les configurations des poids du réseau et dont les transitions sont gouvernées par la présentation aléatoire des entrées. Afin d'appréhender l'évolution de ce processus, on considère la distribution d'un ensemble de processus de ce type dont on étudie l'évolution au cours du temps. Si l'on fait l'hypothèse d'un faible pas d'apprentissage, on peut mettre en évidence les conditions du processus.

Les conditions suivantes sont nécessaires et suffisantes pour garantir avec une probabilité de 1, la convergence vers un état d'équilibre pour tout état initial suffisamment proche de cet état d'équilibre :

$$\lim_{t\to\infty}\varepsilon(t)=0$$

$$\int_0^{\infty}\varepsilon(t)dt=\infty$$

Le pas d'apprentissage $\varepsilon(t)$ doit tendre vers 0 quand t tend vers l'infini pour assurer la stabilisation du processus. Il doit cependant

rester suffisamment important pour mettre une adaptation de la carte aux données, conduisant à une configuration d'équilibre.

Une famille de fonctions $\varepsilon(t)$ satisfaisant ces conditions est donnée par :

$$\varepsilon(t)=t^{-\alpha} \quad avec \quad 0<\alpha\leq 1$$

3.5.2 Etat d'équilibre

Le problème de l'auto – organisation des poids des neurones est un problème complexe qui reçoit une réponse précise si l'on travaille dans un espace de dimension 1 ou 2. Dans tous les autres cas, les multiples exemples présentés dans la littérature tendent à montrer que le comportement est le même. Nous citons la proposition donnée par Kohonen quand l'espace des entrées est de dimension 1 et la carte est de dimension 1 :

Proposition [46]

En dimension 1, à partir d'une initialisation aléatoire des poids W_j (où j parcourt tous les neurones de la carte), pour un t grand ($t \to \infty$) l'ensemble des vecteurs de poids s'organise d'une façon ordonnée dans une séquence croissante où décroissante. Par ailleurs, pour t ayant une valeur importante, la fonction densité des poids W_j approxime la distribution des entrées $p(z)$.

L'évolution du système est un processus de Markov. L'état d'équilibre est un état absorbant et correspond à l'état ordonné des poids. La configuration d'équilibre atteinte par le processus, permettra de mesurer l'adéquation de la carte avec la distribution de probabilité de l'entrée $p(z)$. La carte représente alors une image ordonnée de la fonction densité de l'entrée $p(z)$.

3.5.3 Voisinage

On peut modéliser une version continue de la fonction indicatrice sous forme d'une fonction plus proche du chapeau mexicain. On adopte le plus souvent une fonction gaussienne (FIG 3.11). On peut remarquer que l'on néglige l'existence de relations inhibitrices présentes dans la fonction chapeau mexicain. Elles seront présentes de façon implicite par l'intermédiaire de l'algorithme d'apprentissage.

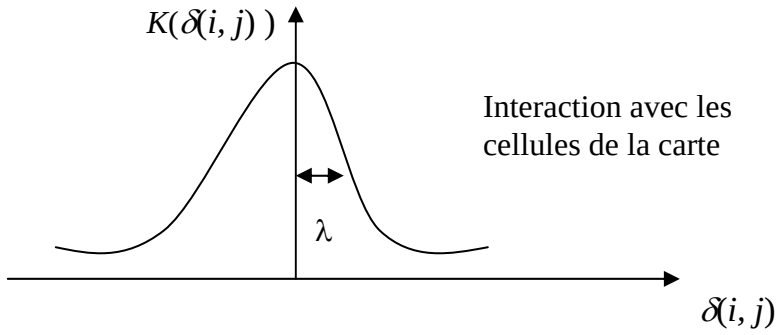


FIG 3.11. ***FONCTION DE VOISINAGE DE TYPE GAUSSIEN (D'APRES [29],P20)***

Cette fonction de voisinage s'écrit à l'aide de la distance sur la carte $\delta(i, j)$:

$$K(\delta(i, j)) = exp(- \frac{\delta^2(i,j)}{\lambda^2}) \quad où\ \lambda\ est\ l'écart\ type$$

Cette fonction détermine le degré d'influence du neurone *j* sur le neurone *i*. La valeur de cette fonction pour *i* dépend de la distance entre *i* et *j* définie sur la carte.

L'utilisation d'une gaussienne pour pondérer le voisinage introduit pour chaque neurone un voisinage global. La taille réelle de ce voisinage est en fait limitée par l'écart type λ de la gaussienne. Les cellules se trouvant au delà de cette étendue ont une influence négligeable mais non nulle sur la cellule *i*.

On pourrait faire évoluer ce voisinage au cours du temps en considérant l'étendue $\lambda(t)$ décroît en fonction du temps *t*. Si l'on considère que l'étendue $\lambda(t)$ décroît en fonction de *t*, la fonction voisinage aura la même forme avec un écart type décroissant quand *t* varie. La FIG 3.12 représente l'influence du voisinage pour une étendue $\lambda(t)$ décroissante au cours du temps.

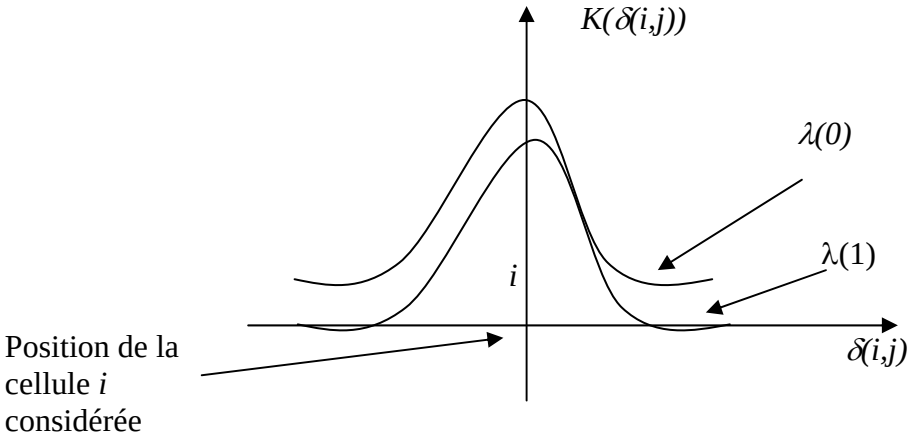


FIG 3.12. *EVOLUTION DE LA TAILLE DU VOISINAGE ET D'INTERACTION ENTRE LA CELLULE I ET SES VOISINS EN FONCTION DU TEMPS (D'APRES [29], P20).*

On choisit en général pour ce type de voisinage une décroissance exponentielle pour la fonction écart type $\lambda(t)$.

Pour tenir compte de ce voisinage au cours de l'apprentissage, on modifie la règle de mise à jour des poids par :

$$W_j^t = W_j^{t-1} + \varepsilon(t)K(\delta(c,j))(W_j^{t-1}-z)$$

3.5.4 Une variante sur les cartes topologiques

En général, les cartes topologiques s'appliquent à des données structurées et organisées sous formes de tableaux : Individus/variables , où les variables sont fixes pour tous les exemples à traiter. Lors de l'apprentissage, les neurones des vecteurs d'entrées sont forcées aux valeurs des variables définies.

Nous voudrions traiter le cas où un individu – étant caractérisé par *p* variables – n'est pas présenté directement au réseau. L'ensemble de ses variables est présenté au réseau par succession de tranches, dont la présence est redondante dans le temps. Ce qui explique le changement du comportement si il a lieu. Cela rend le codage du vecteur d'entrée dynamique.

Nous proposons d'effectuer une variante sur la modification des poids dans l'algorithme de Kohonen en ajoutant une fonction « d'élection » qui permet de masquer les neurones qui ne participent pas à l'élection du neurone dans la carte.

Nous présentons un exemple de codage des données :

Soit *i* un individu caractérisé par *p* variables

La forme de cet individu est présentée sous forme d'une session à temps discret. En stationnarisant la forme dans le temps, nous sélectionnons des caractéristiques partielles à chaque intervalle de temps. Ces caractéristiques sont présentées au réseau et ne mettent en considération que les poids qui leur sont associées.

Session *i* à *p* variables

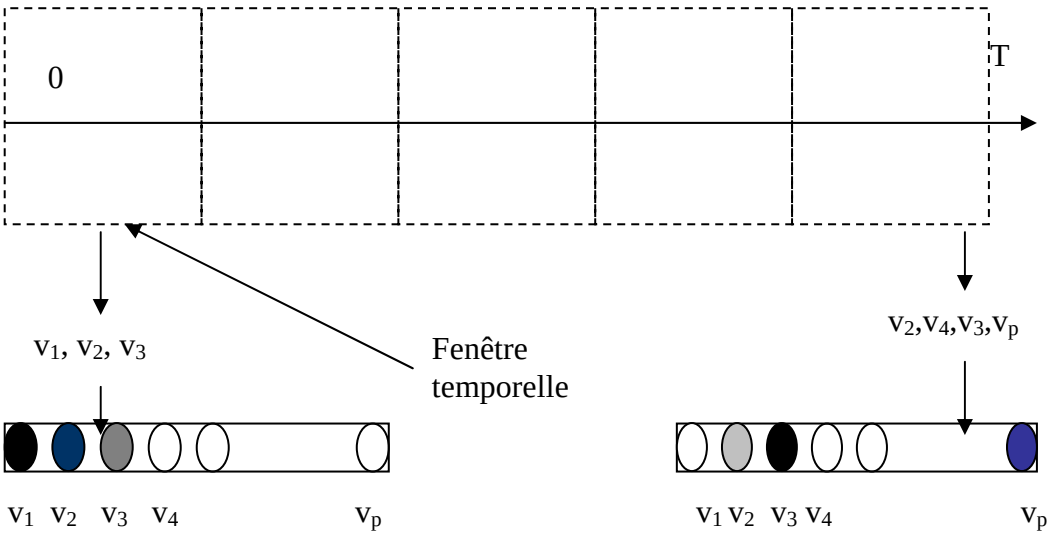


FIG 3.13. *CODAGE PARTIEL DES DONNEES DANS LE TEMPS.*

A chaque intervalle de temps, nous n'extrayons qu'une portion de caractéristiques sur notre individu, nous sélectionnons les neurones associés à ses caractéristiques[1], et puis nous présentons le contenu de la fenêtre à la carte de Kohonen en ne prenant en compte que les neurones qui participent à l'élection (FIG 3.13).

On peut remarquer une différence de couleur sur les neurones de la même fenêtre, cela est dû aux différences des valeurs des variables dans les données et dans notre application cela explicite l'intérêt qu'apporte chaque page sur l'utilisateur pendant sa navigation dans le site.

[1] En général, le codage se fait en forçant les neurones aux valeurs des variables qui leur sont associées dans la fenêtre temporelle.

Nous présentons la formulation algorithmique qui interprète cette nouvelle version :

1. **Initialisation :** $t = 0$, initialiser les poids W_j, à des valeurs tirées aléatoirement

Au temps t , Présenter la forme z sur les i caractéristiques , $i \in F$ (fenêtre temporelle) et faire :

2. **Compétition :** choix du gagnant W_c^t :

$$\sum_{i \in F} (z_i - W_{i,c}^t)^2 = \min_{j \in C} [\sum_{i \in F} (z_i - W_{i,j}^t)^2]$$

3. **Adaptation :** mise à jour des poids dans le voisinage du gagnant :

$$W_{i,j}^t = W_{i,j}^{t-1} + \varepsilon(t) \; \mathbf{e(i)} \; K(\delta(c,j))(z - W_{i,j}^{t-1})$$

$$\text{avec } e(i) = \begin{cases} 1 \; si \; i \in F \\ 0 \; sinon \end{cases}$$

Au temps t+1, glisser F sur le corpus d'apprentissage, sélectionner les neurones i tels que $i \in F$ et répéter les deux étapes de compétition et d'adaptation.

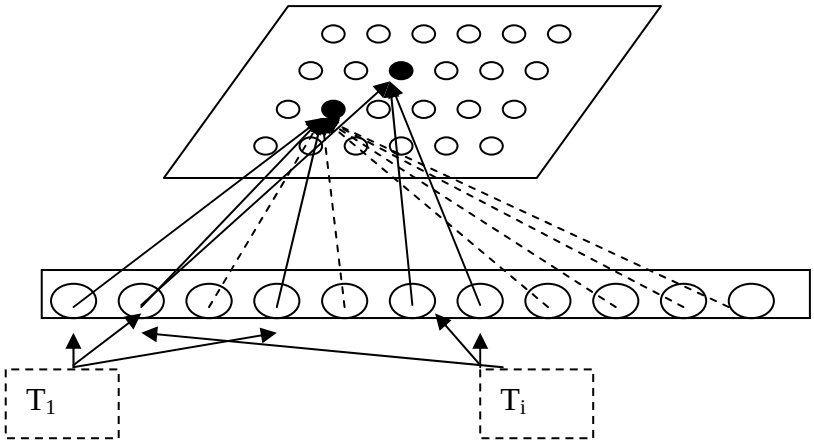


FIG 3.14. ***CARTE DE KOHONEN AVEC ELECTION DES NEURONES DE LA CARTE DEPENDANT DES NEURONES PARTICIPANT A L'ELECTION DANS CHAQUE PAS DU TEMPS.***

Concrètement chaque individu représente un ensemble de fenêtres temporelles contenant chacune un sous ensemble de variables. Chaque fenêtre fournit ses variables à la carte de Kohonen pour élire un neurone. A la fin de la session, l'individu sera donc représenté par un ensemble de neurones gagnants sur la carte.

A la fin de l'apprentissage, nous aurons une classification de *portions* d'individus, i.e. qu'il peut être mis dans une classe *i* suivant un sous – ensemble de variables donné, et dans une autre classe *j* suivant un autre sous – ensemble de variables. Pour notre application, ce stimulus décrit explicitement le changement de comportement que peut avoir un internaute naviguant dans un site web. Si l'individu apparaît dans plusieurs classes, nous pourrons calculer la distances entre ces classes pour évaluer son changement de comportement.

3.6 De la théorie à la pratique

3.6.1 Choix des paramètres

Un bon choix du pas d'apprentissage $\varepsilon(t)$ permet d'obtenir un état d'équilibre du système. Cependant l'aspect stochastique de l'algorithme peut faire tendre le système vers un « mauvais » état d'équilibre. Par exemple si le pas d'apprentissage diminue trop vite, d'une part la distribution n'est pas couverte et d'autre part la topologie n'est pas préservée. L'algorithme original de Kohonen ne prévoit pas une forme spécifique du pas du gradient $\varepsilon(t) = \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{\frac{t}{t\max}}$ où ε_i et ε_f sont respectivement le pas d'apprentissage initial et final et t et t_{max} sont respectivement le temps et le nombre maximal du temps.

De même, la décroissance de la fonction $\lambda(t)$ qui maîtrise la taille du voisinage a un effet déterminant sur la solution obtenue à la convergence.

Dans ce paragraphe nous nous intéressons plus particulièrement au pas d'apprentissage. Le choix de ce paramètre ne va pas sans influer sur la configuration finale de la carte **[29]**.

3.6.2 Mesure de la qualité de quantification

Les méthodes de visualisation ont montré leur capacité à permettre une évaluation simple des résultats. En revanche il est nécessaire de trouver d'autres méthodes pour évaluer quantitativement les

représentations fournies par les cartes topologiques de l'algorithme SOM. Nous avons vu que contrairement à d'autres méthodologies de représentation de données, le modèle SOM, tel qu'il a été formulé par Kohonen, n'offre pas d'objectif analytique clair et précis. Par contre, nous connaissons les conditions que doit remplir une carte SOM pour être jugée efficace sur un ensemble d'entrées donné. Parmi ces conditions nous pouvons citer :

- La minimisation de l'erreur de discrétisation appelée distorsion : il faut que les distances d'un vecteur de poids aux entrées qui l'activent soient minimales.
- La représentation de manière correcte de la distribution des entrées : il faut que les vecteurs poids approximent la densité de probabilité des vecteurs d'entrées. Ce critère est nommé erreur ou contraste.

3.6.2.1 Distorsion

Pour une entrée z de la base A nous appelons erreur de distorsion, la distance Euclidienne entre z et le vecteur poids de la cellule gagnante c :

$$q(z) = \|z - W_c\|^2$$

L'ensemble des vecteurs poids minimise l'erreur de distorsion totale lorsque la moyenne Q des $q(z)$ est minimale :

$$Q(z) = \frac{1}{N}\sum_z q(z)$$

3.6.2.2 Contraste

La carte doit représenter la densité de probabilité des entrées au plus juste : les zones à forte densité seront représentées par plus de cellules que celles à faible densité. Autrement dit, les cellules doivent être réparties de manière à ce que chacune représente le même nombre d'entrées.

Si B_i, est l'ensemble des formes z de A pour lesquelles la cellule i est gagnante :

$$B_i = \{z \in A / \|z - W_i\|^2 = \min_{c \in C} \|z - W_c\|^2 \}$$

Pour avoir une bonne représentation de la distribution d'entrée, il faut que, pour deux cellules quelconques i et j, on ait : $card(B_i) \approx card(B_j)$

Si on pose $n_i = card(B_i)$ *et* $k = card(C)$, le contraste entre les différentes cellules est évalué par :

$$CNT = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left\{\frac{ni-\bar{n}}{\bar{n}}\right\}^2} \text{ avec } \bar{n}=\frac{1}{k}\sum_{i}^{k} n_i$$

Cette quantité doit tendre vers une valeur limite en fin d'apprentissage, valeur que l'on souhaite proche de 0.

Les deux critères précédents : distorsion et contraste ne constituent pas un objectif en soi dans la mise au point de la carte apprise par l'algorithme SOM. Ils sont utilisés pour vérifier et juger la qualité d'une carte au cours de l'apprentissage. On retiendra alors empiriquement la carte qui minimise au mieux la distorsion ou le contraste.

Nous allons présenter dans la section suivante, une méthode de classification hiérarchique pour un but d'optimisation du nombres de neurones de la carte obtenu par SOM.

3.7 Choix du nombre de classes

Plusieurs méthodes de partitionnement fournissent une solution dépendant de la configuration et du nombre de classes. Une littérature très abondante existe autour de l'initialisation, mais laisse souvent à l'utilisateur le problème du choix du nombre de classes **[30]**.

3.7.1 Hiérarchie et distance ultramétrique

En partant d'un tableau de données *E,* on calcule une mesure de proximité entre ses individus. Plus la valeur de cette mesure est petite plus les individus doivent être semblables. Cette mesure de proximité entre ces individus est caractérisée par un indice de distance ou une distance sur l'espace des données, soit *Dom*.

Cette distance d vérifie les propriétés suivantes :

- $d(x, y)=0 \Leftrightarrow x=y$
- $\forall x, y \in Dom\ d(x, y)=d(y, x)$ (symétrie)
- $\forall x, y, z \in Dom\ d(x, z) \leq d(x,z)+ d(y,z)$ (inégalité triangulaire)

Un indice de distance ne vérifie que les deux premières propriétés. Les méthodes de classification hiérarchique ont pour objectif de construire une suite de partitions emboîtées appelée hiérarchie. La

représentation graphique de ces hiérarchies se fait par un arbre hiérarchique ou dendrogramme.

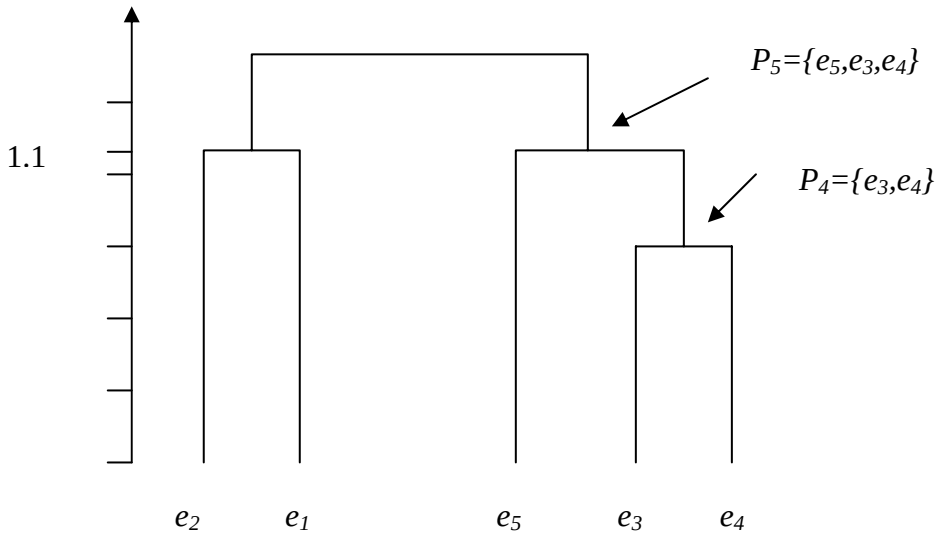


FIG 3.15. ***UN DENDROGRAMME (D'APRES [30], P185).***

La proximité entre l'individu $\{e_5\}$ et le palier $P_4=\{e_3,e_4\}$ est représentée par le niveau du palier $P_5=\{e_5, e_3, e_4\}$et sa valeur est égale à 1.1 (FIG 3.15). Avec ce graphique on détermine toutes les proximités δ entre deux éléments quelconques de la hiérarchie. Cette distance δ possède la propriété suivante :

$$\forall h_1, h_2, h_3 \in H \text{ on a } \delta(h_1, h_2) \leq max\{\delta(h_1, h_3), \delta(h_2, h_3)\}$$

Cette inégalité, appelée l'inégalité ultramétrique, est plus contraignante que l'inégalité triangulaire. De ce fait, il faut ajouter à la distance initiale la propriété suivante :

$$\forall x, y, z \in Dom\ d(x, y) \leq Max\{d(x,z), d(z,y)\}\ (inégalité\ ultramétrique)$$

L'objectif de toute méthode de classification hiérarchique, est de construire une ultramétrique δ, la plus proche de la distance initiale d au sens d'un critère défini a priori.

La bijection entre les hiérarchies indicées ou dendrogramme et les ultramétriques permet la construction de cette ultramétrique en

utilisant les algorithmes de construction de dendrogrammes. Deux stratégies de construction de ces dendrogrammes sont proposées :

- On construit le dendrogramme à partir du bas c'est à dire à partir des feuilles terminales de l'arbre. Dans ce cas, on agrège deux par deux, les classes les plus proches. De proche en proche ce processus est utilisé jusqu'à l'obtention d'une seule classe.
- On construit le dendrogramme à partir du haut c'est à dire en procédant par division successive de l'ensemble *E* jusqu'à obtenir un seul individu par classe. On obtient ainsi les feuilles terminales de l'arbre.

La première stratégie est la plus utilisée car elle est de moindre complexité que la seconde et sa mise en œuvre est aisée grâce à l'algorithme de la classification ascendante hiérarchique, dit de la CAH qui sera présenté dans le paragraphe suivant.

La mise en œuvre de cette seconde stratégie entraîne des problèmes d'optimisation combinatoire, car l'ensemble des partitions de deux classes est très grand et actuellement, il n'y a pas de propriétés mathématiques permettant d'éviter l'énumération de toutes ces partitions en deux classes.

3.7.2 Les mesures d'agrégation entre les classes

Pour mettre en œuvre l'algorithme de la CAH, il faut construire une mesure d'agrégation *D* entre les classes. Les trois mesures d'agrégation entre deux classes *A* et *B*, présentées ci-après, sont les plus utilisées.

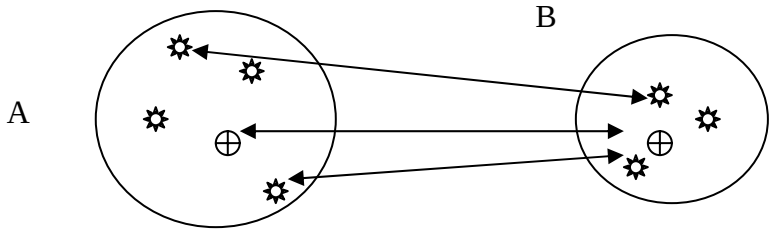


FIG 3.16. ***MESURES D'AGREGATION ENTRE DEUX CLASSES (D'APRES [30], P187).***

- La mesure d'agrégation du lien minimum entre les classes *A* et *B*

$$D_1(A, B) = \min_{a \in A, b \in B} d(a,b)$$

- La mesure d'agrégation du lien maximum entre les classes *A* et *B*

$$D_2(A,B) = \max_{a \in A, b \in B} d(a,b)$$

- La mesure de l'augmentation de l'inertie ou l'indice de WARD entre les classes *A* et *B*

$$D_3(A,B) = \frac{\mu(A)\mu(B)}{\mu(A)+\mu(B)} d^2(g_A, g_B)$$

où g_A est le centre de gravité de la classe *A* et où la fonction μ correspond à la pondération des classes qui est souvent égale à l'effectif de la classe.

3.7.3 Algorithme général de la classification ascendante hiérarchique

Cet algorithme, nommé CAH, consiste à construire à l'aide d'une mesure d'agrégation (i.e. une distance entre les parties de E) une suite de partitions emboîtées à partir de la partition la plus fine, celle contenant un individu par classe, jusqu'à obtenir la partition la plus grossière, celle ne contenant qu'une seule classe. Il s'énonce de la manière suivante.

- Initialisation

On se donne au départ, une partition $Q^{(0)}$ constituée de *N* classes telle que :

$$Q^{(0)} = (Q_1^{(0)}, \ldots Q_N^{(0)}) \text{ avec } Q_i^{(0)} = \{e_i\}.$$

On se donne une mesure d'agrégation $D : P(E) \times P(E) \rightarrow R^+$ qui doit vérifier la condition suivante :

$$\forall e_i, e_m \in E, D(\{e_i\}, \{e_m\}) = d(e_i, e_m)$$

- Etape agrégative

Construire une nouvelle partition $Q^{(N-K)}$ contenant *K* classes à partir de la partition $Q^{(N-K-1)}$ et les plus proches *K*+1 classes au sens de la mesure d'agrégation *D*. En cas d'égalité on choisit la première solution rencontrée. Ceci nous permet d'avoir l'unicité de la hiérarchie ainsi construite et cette hiérarchie est toujours binaire.

- recommencer l'étape agrégative jusqu'à obtenir une seule classe, c'est à dire la partition grossière.

3.7.4 Construction de la hiérarchie indicée

Une hiérarchie indicée est un couple (H, f) où H est une hiérarchie et f une fonction réelle sur l'ensemble $P(E)$ telle que :

- $f(H) = 0 \Leftrightarrow |h| = 1$
- $\forall\ h_1, h_2 \in H$ si $h_1 \subset h_2$ alors on a $f(h_1) < f(h_2)$

Le dendrogramme, associé à cette hiérarchie indicée, est dit inversion. La relation souvent utilisée entre cette fonction f et une mesure d'agrégation D entre classes est la suivante :

$$\forall h_1, h_2 \in H,\ f(h_1 \cup h_2) = D(h_1, h_2)$$

Pour nos trois mesures d'agrégations définies au paragraphe précédent cette relation permet toujours de construire une hiérarchie indicée. Par contre si la mesure d'agrégation est la distance entre les deux centres de gravité, c'est à dire :

$$D_4(A, B) = d^2(g_A, g_B)$$

Alors le couple (H, f) n'est pas toujours une hiérarchie indicée car des inversions sont possibles (i.e. il peut exister des classes h_1 et h_2 ne vérifiant pas la condition suivante :

$$\forall h_1, h_2 \in H\ f(h_1 \cup h_2) = Max\ \{D(h_1, h_2), f(h_1), f(h_2)\}$$

Il faut noter que cette relation donne les mêmes résultats que la relation précédente si la hiérarchie indicée n'a pas d'inversions. A partir de cette hiérarchie indicée, on définie, de la manière suivante, une distance ultramétrique δ entre les individus :

$$\delta(e_i\ e_m)\ \min_{h \in H} \{f(h) / e_i \in h \text{ et } e_m \in h\}$$

De manière plus constructive, il faut lors de l'agrégation des deux paliers h_1, h_2, mettre à jour le tableau de la distance ultramétrique, i.e. :

$$\forall e_i \in h_1, e_j \in h_2 \text{ on a } \delta(e_i, e_j) = D(h_1, h_2) = f(h_1 \cup h_2)$$

Donc, plus le palier regroupant un ensemble d'individus se trouve dans le bas de l'arbre, plus la valeur de l'ultramétrique δ entre les individus de cet ensemble est petite.

3.7.5 La formule de récurrence de Lance et de Williams

Lors de l'étape agrégative de l'algorithme de la CAH, on regroupe deux classes de la partition. Il est nécessaire de recalculer la mesure d'agrégation entre la classe ainsi formée et les autres classes de la partition.

Lance et Williams en 1967 ont proposé, lors du regroupement des deux classes *A* et *B* en *C*, la formule de récurrence suivante :

$$A, B, C \in P(E)\ \ D(C, A\cup B) = \alpha_1 D(C, A) + \alpha_2 D(C, B) + \alpha_3 D(A, B) + \alpha_4 |D(C, A) - D(C, B)|$$

Nous pouvons vérifier rapidement que les coefficients α_1, α_2, α_3, α_4 sont :

- pour la mesure d'agrégation du lien minimum égaux à :

$$\alpha_1 = \alpha_2 = -1/2\ ,\ \alpha_3 = 0,\ \alpha_4 = -1/2$$

- pour la mesure d'agrégation du lien maximum égaux à :

$$\alpha_1 = \alpha_2 = 1/2\ ,\ \alpha_3 = 0,\ \alpha_4 = -1/2$$

- et pour la mesure de l'augmentation de l'inertie ou l'indice de WARD ils sont égaux à

$$\alpha_1 = \frac{\mu(C)+\mu(B)}{\mu(F)}\ \alpha_2 = \frac{\mu(C)+\mu(A)}{\mu(F)}\ ,\ \alpha_3 = \frac{\mu(C)}{\mu(F)},\ \alpha_4 = 0 \text{ avec } F = A\cup B\cup C.$$

3.8 Application : visualisation du site tel que les internautes l'utilisent

L'application 2.8 nous a fournit un tableau de données de 1260 × 82 éléments. Ce tableau décrit une relation partielle entre toute Url et les autres Urls du fichier Log. En effet, chaque Url est caractérisée par un vecteur d'Urls : une partie qui décrit la précédence et l'autre décrit la succession.

Malgré sa taille réduite et simplifiée, le tableau d'Urls reste assez complexe à interpréter. Cette complexité est due à la relation et au nombre important des exemples composant les données. Pour cette raison, nous proposons d'appliquer une carte topologique de kohonen. Cette carte a pour objectif de résumer de manière simple un ensemble de données multi – dimensionnelles.

3.8.1 Dimensions de la carte

La forme de la maille à la base du réseau est le plus souvent rectangulaire ou hexagonale. Un maillage rectangulaire est particulièrement adapté pour des applications nécessitant la visualisations des données traitées.

Pour déterminer le nombre des neurones de la carte, l'heuristique de Kohonen donnerait un nombre C = 177 neurones.

Avec notre heuristique (décrite dans §3.4.3), C=64 neurones.

Les dimensions de la cartes x, y :

x = 7, y = 9.

3.8.2 Construction de la carte

Nous appliquons l'algorithme des cartes topologiques de Kohonen (SOM : §3.5.4) sur la matrice quasi-comportementale dynamique pour le regroupement des pages similaires (FIG 3.17)

Souscription Reserve	Souscription Aiguillage	Fiche technique Auto	Fiche Technique perso	Fiche technique Reserve	oui je veux l'assurance crédit Auto
Souscription Perso			Pieces justificatives	Fiche technique Auto	Simulations
Souscription Auto	Les differents credits	Loi Neiertz	Comprendre le taux du credit	Produit travaux	Produit travaux
		Dialoguez avec nous	Rembourcement anticipé	Plan du site	Liste de produit
Devenez membre		Dialoguez avec nous	Info Credit	Analyse	Plan du site
Mot de passe oublié		Plan du site	La foire aux questions	Analyser votre capacite d'emprunt	Utilisez les simulateurs
		Les infos pour obtenir votre credit	La foire aux questions	Qui somme nous	Les conditions d'accès au credit
	Les differents credits	Loi Neiertz	Pieces justificatives		Interet du credit
Entree	Accueil	Pieces justificatives	Pieces justificatives	Comprendre le taux du credit	Sortie

FIG 3.17 ***CARTOGRAPHIE DU SITE WWW.123CREDIT.COM***

La carte construite donne lieu à un paquet de classes structuré en topologie prédéfinie. Une CAH (§3.7.3) est appliquée sur les prototypes représentant les neurones pour optimiser le nombre de classes représentées par la carte.

Les zones obtenues par la CAH ont besoin d'être étiquetées. Cependant lors de l'utilisation de la carte, chaque Url peut élire plus d'un neurone. Cela est dû à son changement d'importance de passage d'un jour à l'autre. Intuitivement, si la carte est bien faite, les différents neurones élus par la même Url doivent être assez proches.

Pratiquement, pour étiqueter la carte dans ce cas, on procède par vote majoritaire

Le principe du vote majoritaire consiste à caractériser chaque neurone par l'Url qui représente le plus grand pourcentage d'activation. En d'autres termes, après utilisation de la carte chaque neurone (classe) contiendra un ensemble d'Urls et comme chaque Url possède plusieurs exemples, alors chacune peut élire autant de fois le neurone associé. Par conséquent, pour choisir l'Url qui doit représenter cette classe, on procède par vote majoritaire, i.e. choisir celle ayant le plus grand nombre d'élection par rapport à l'effectif de la classe.

Remarque :

Si le neurone contient plusieurs Urls similaires en pourcentage de présence, on choisit celle qui est la plus proche de lui en terme de la distance euclidienne.

La deuxième étape consiste à attribuer des libellées pour chaque zone (ensemble de neurones proches dans la carte). L'Url « étiquette » qui caractérise la carte sera celle qui sera omniprésente dans la zone de la même manière que l'étiquetage d'un neurone (FIG 3.18).

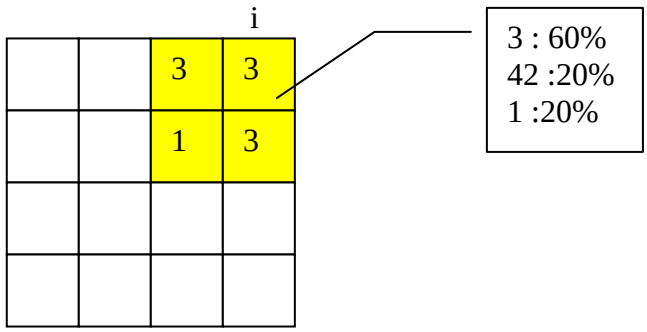


FIG 3.18. ***PRINCIPE D'ETIQUETAGE DE LA CARTE PAR VOTE MAJORITAIRE : LE NEURONE I EST ETIQUETE PAR L'URL 3 CAR ELLE REPRESENTE 60% DE PRESENCE PAR RAPPORT AUX AUTRES URLS (42 ET 1) ET LA ZONE COLOREE EST ETIQUETEE PAR CETTE MEME URL CAR ELLE EST PRESENTE DANS LA PLUS PART DES NEURONES***

Nous pouvons remarquer que L'Url 3 dans la figure ci-dessus a activé plusieurs neurones. Cela est dû à sa présence dans la base sous forme de plusieurs exemples.

Nous avons appliqué cette démarche au site : www.123credit.com et on a obtenu la carte suivante :

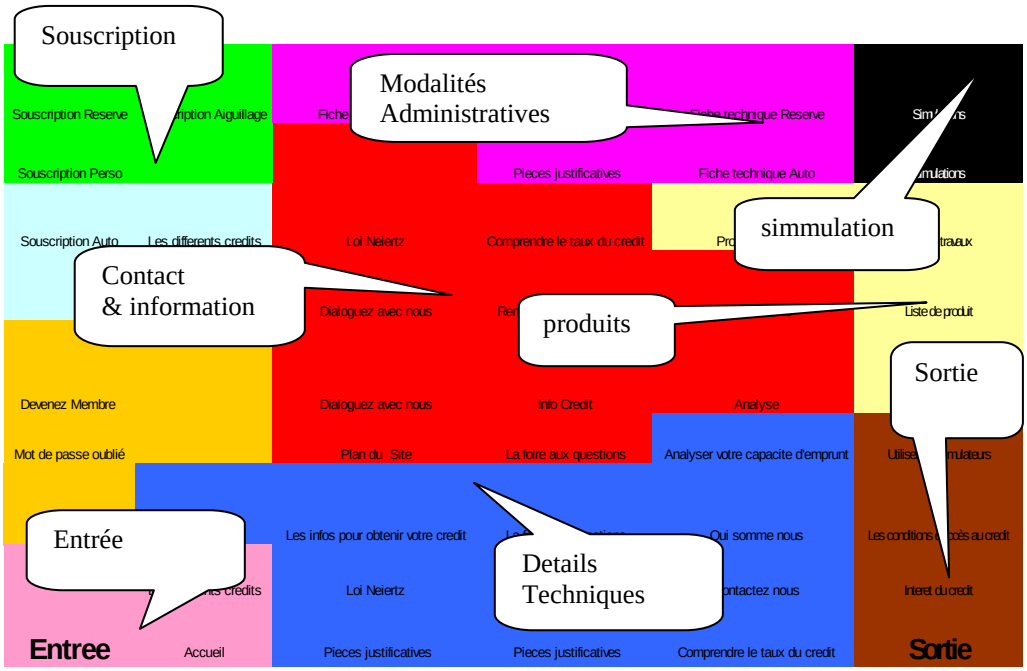


FIG 3.19. ***CARTOGRAPHIE DU SITE (WWW.123CREDIT.COM) APRES CAH ET ÉTIQUETAGE PAR VOTE MAJORITAIRE***

Commentaire :

Les neurones sont numérotés de haut en bas et de gauche à droite : ainsi, le neurone 0 (étiqueté « souscription réserve») est en haut à gauche de la carte, le 8 (« Entrée ») est en bas à gauche, le 9 (« Souscription Aiguillage ») est à droite du 0, le 62 (étiqueté « Sortie ») est en bas à droite etc.

En appliquant la méthode du vote majoritaire avec la collaboration des experts, nous avons pu attribué à chaque zone colorée de la carte un concept lié à un ensemble de page.

La carte de la figure (FIG 3.19) établit le plan du site www.123credit.com tel que les internautes l'utilisent, dans le sens où une proximité entre deux zones (chaque zone correspondant à une Url ou un groupe d'Urls du site) de la carte correspond à des déplacements fréquents des utilisateurs d'une zone vers l'autre.

La cartographie construite servira de base de projection pour les sessions enregistrées dans le fichier Log. Nous verrons dans le chapitre suivant, le comportement du trafic comportemental qui définit les différents profils des internautes.

3.9 Conclusion

Nous avons défini dans ce chapitre, l'auto – organisation dans les réseaux neuronaux par les caractéristiques liées à la méthode d'adaptation. Nous avons expliqué par la suite, comment l'apprentissage s'effectue dans ces systèmes. Une variante sur l'algorithme des cartes topologiques a été introduite, pour la classification non supervisée des données par portions de variables.

Enfin, le formalisme de la classification hiérarchique a été expliqué en dernier lieu.

Chapitre IV

4 Typologie de sessions : Profiling

Dans ce chapitre, nous proposons une nouvelle approche de codage de sessions (données basées sur des séquences). Nous étudions par la suite l'impact de la quantification vectorielle sur de telles données. Par la suite, nous analysons le comportement de la programmation dynamique sur la reconnaissance des sessions de navigation. Ce processus d'analyse nous permettra de catégoriser les différents comportements d'internautes étant intéressés par un ensemble de catégorie de pages dans un site web.

4.1 Introduction

Les sessions des individus dans la plupart des ensembles de données sont variables selon leurs formes. Le nombre important des sessions nécessite une représentation dans un espace réduit. Cela permet de bien les comprendre. En d'autres termes, nous cherchons à déterminer une typologie de sessions. Pratiquement, nous allons décrire deux méthodes de codage que nous avons développé pour ce genre de formes.

Jusqu'à présent, nous avons pu résumer l'ensemble des pages visitées dans une cartographie simple et optimale. Cette cartographie nous a montré le plan du site tel que les visiteurs le perçoivent. Nous pouvons récapituler ce processus dans la figure suivante :

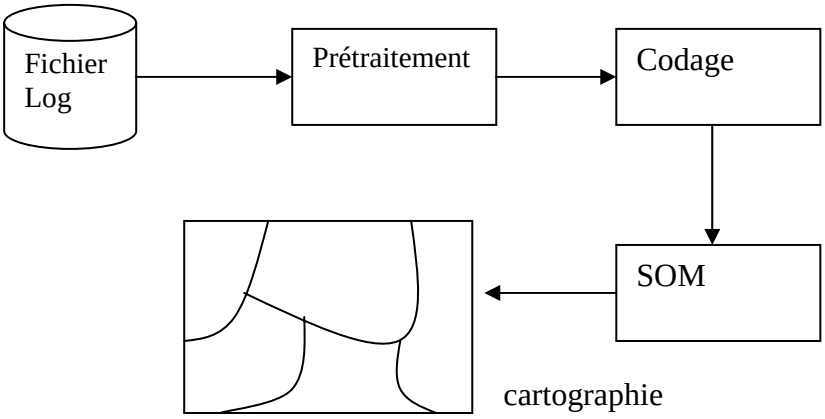


FIG 4.1. ***PROCESSUS GENERAL DE LA CONSTRUCTION DE LA CARTOGRAPHIE D'UN SITE***

Après avoir construis la cartographie du site, nous pouvons visualiser les différentes transactions du fichier Log en les projetant sur cette cartographie. Une session de navigation devient donc, une suite de neurones où chacun représente une classe de pages similaires en terme de navigation.

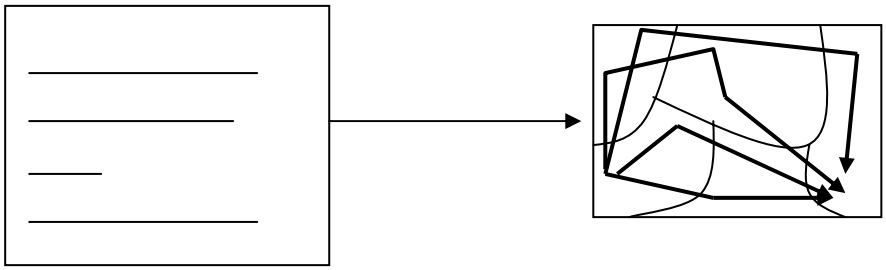


FIG 4.2. ***PROJECTION DU TRAFIC DU FICHIER LOG SUR LA CARTE DE KOHONEN***

Il faut mentionner qu'à cause du nombre important des sessions du Log, leur représentation sur la cartographie devient invisible voire incompréhensible. Pour cette raison, nous voudrions réduire l'espace du trafic et le remplacer par un espace de chemins prototypes.

4.2 Les données

La typologie de sessions nécessite un tableau de données dont :

les individus : sont les sessions

les variables : sont les symboles de séquences composant les sessions.

Exemple :

$$S_i : p_i\ p_j\ p_k \ldots$$

$$S_{i+1} : p_c\ p_j \ldots$$

La variabilité de la longueur des séquences implique un vrai problème de représentation de données (problème de décalage). Les vecteurs décrivant le trafic sont donc de longueurs différentes.

Pour résoudre ce problème, nous proposons une nouvelle méthode de codage. Cette méthode consiste à transformer les sessions en vecteurs de même taille.

4.2.1 Segmentation

Globalement, les sessions extraites du fichier Log représentent trois grandes classes de sessions :

- Les sessions « Achat » : reconnues par des Urls particulières appelées Urls d'achat
- Les sessions « Non-Achat Long » (NAL) : Ce sont des sessions qui ne contiennent pas des URLs d'achat mais leurs temps dépassent un seuil fixé empiriquement.
- Les sessions « Non-Achat Court » (NAC): Ce sont des sessions « Non- Achat » qui ne dépassent pas le seuil fixé.

Au lieu de représenter les sessions par les successions de pages visitées, nous les représentons par la succession de classes de pages (neurones) issues de la classification non supervisée (cf FIG 4.3).

Identificateur de session	Indicateur d'achat	Page	Temps
1	Achat	1	22"
1	Achat	2	43"
1	Achat	4	12"
1	Achat	23	10"
1	Achat	6	13"
2	N-Achat Court	1	43"
2	N-Achat Court	2	76"
2	N-Achat Court	7	32"
3	N-Achat Long	1	50"
3	N-Achat Long	3	23"
3	N-Achat Long	4	6"
3	N-Achat Long	45	23"
3	N-Achat Long	3	40"

Codage

Identificateur de session	Indicateur d'achat	Neurone	Temps
1	Achat	42	22"
1	Achat	3	43"
1	Achat	3	12"
1	Achat	6	10"
1	Achat	7	13"
2	N-Achat Court	42	43"
2	N-Achat Court	3	76"
2	N-Achat Court	8	32"
3	N-Achat Long	42	50"
3	N-Achat Long	4	23"
3	N-Achat Long	3	6"
3	N-Achat Long	8	23"
3	N-Achat Long	4	40"

FIG 4.3. *CODAGE DES URLS APRES UTILISATION DE LA CARTE*

Ce codage réduit la variété du trafic mais ne résout pas le problème de la variabilité de la longueur.

4.3 Codage par poids de position (Pdp)

Pour remédier au problème de décalage évoqué dans §4.2, nous proposons une méthode de codage qui tient compte de la position de la station[1] et de sa fréquence dans la session.

Le principe consiste à attribuer un poids sur la station selon sa position dans la session : Plus une station est loin dans une session, plus son poids est important.

La fréquence est une variable liée à chaque station pour déterminer le nombre de passage par la station.

Nous présentons donc, notre algorithme comme suit:

// Initialisation

Pour i = 1 à N Faire

Poids(i) = 0

Frequence(i)=0

FinPour

// poids de position et fréquences

Pour i ∈ session do

Si i = 1ère URL de la session Alors

Poids(i) = α *//α : est un paramètre initial*

FinSi

[1] La station ici, représente une page dans la session

Si Fréquence(i) = 0 Alors

Poids(i) = Poids(Précédent(i)) + δ

Sinon

Poids(i) = [Poids(i)+(Poids(Précédent(i)) + δ)]/2

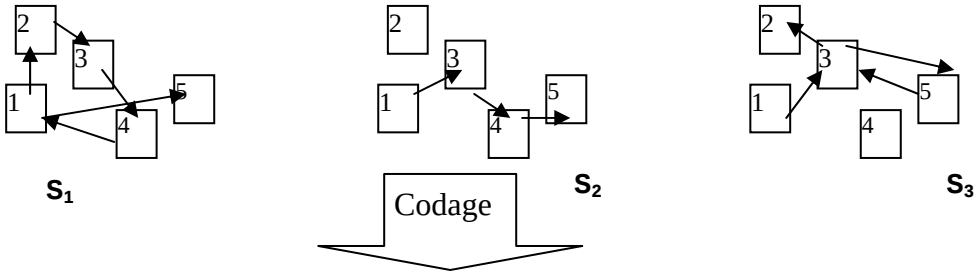
FinSi

Fréquence(i) = Fréquence(i) + 1

FinPour

Tel que:
N: le nombre de neurones de la carte
Précédent(i) : la station qui précède la station i dans le chemin
δ: un coefficient empirique qui détermine la position.

Exemple :



Identificateur de session	Neuone1		Neurone2		Neurone3		Neurone4		Neurone5	
	position	fréquence	position	fréquence	position	fréquence	position	fréquence	position	fréquence
S1	14	2	11	1	12	1	13	1	15	1
S2	10	1	0	0	11	1	12	1	13	0
S3	10	1	14	1	13	2	0	0	12	1

FIG 4.4 ***EXEMPLE DE CODAGE DE CHEMINS PAR LA METHODE DES POIDS DE POSITION AVEC α=10 ET ΔW=1***

La figure FIG 4.4 représente le codage des sessions S_1, S_2 et S_3 en un tableau de vecteurs de même dimension

Par exemple , la session S_1 représentée par la séquence : 1 2 3 4 1 5 est codée comme suit :

Pour le neurone 1 , on affecte le poids de départ fixé à 10, on avance vers 2, donc ce dernier aura le poids 10+1 (11), ensuite le 3 aura le poids encore incrémenté (12), le 4 aura le poids 13, on retourne à 1 donc ce dernier change de poids (14) ainsi que de fréquence (2) et finalement le neurone 5 aura le poids 15.

4.4 Quantification vectorielle

4.4.1 Principe général

La quantification vectorielle (QV) a été introduite pour créer des références statistiquement plus représentatives **[53]**, et en même temps économiques au stockage. L'idée essentielle de cette technique résulte du fait que dans l'espace de représentation de la base des sessions, les vecteurs issus du codage n'occupent que des sous espaces sous forme de nuages.

Soit $X=\{x_1,x_2,\ldots .x_n\}$ *un vecteur aléatoire de l'espace* R^n *muni de la métrique (distance)* d

La QV consiste à coder ce vecteur par un ensemble de prototypes. Ces derniers réalisent un partitionement de l'espace en classes (cf FIG 4.5)

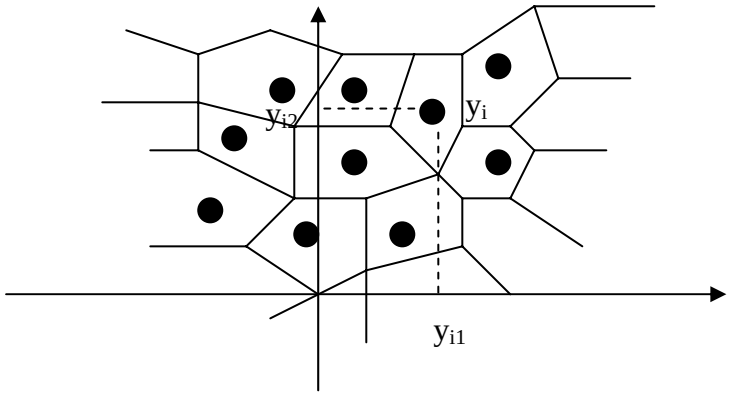


FIG 4.5. ***L'ENSEMBLE DE DIMENSION 2 EST PARTITIONNE EN K CLASSES***

L'espace étant partitionné, le codage est alors simple : tous les points qui se trouvent à l'intérieur d'une classe C_i retrouvant le code y_i. Si l'espace est partagé en k classes, on aura k prototypes où :

$$Y=\{y_1, y_2, \ldots, y_k\}$$

$$y_i=y_{i1},y_{i2},\ldots, y_{in}$$

Y: s'appelle l'ensemble de prototypes ou dictionnaire (*codebook*)

y_i : prototype ou représentant (*codeword*)

En remplaçant X par l'ensemble de prototypes y_i on introduit forcément une erreur dite de quantification (ou de distorsion). Cette erreur peut être mesurée à l'aide d'une distance d. l'information quantifiée est donc plus ou moins bruitée.

En général, si on connaît la distribution multidimensionnelle *p(x)* pour chaque entrée *X*, un quantificateur optimal est celui qui minimise l'espérance mathématique entre l'entrée et la sortie de ce quantificateur. En pratique, cela consiste à minimiser la quantité :

$$D = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} d(xi, Q(x_i))$$

x_i et $Q(x_i)$ sont respectivement, l'entrée et la sortie du quantificateur.

Le problème qui se pose est celui de l'existence de la limite. Si on considère les x_i comme étant des réalisations d'un processus aléatoire *x*, et si ce processus est stationnaire et ergodique **[53]** alors cette limite tendra vers $E(d(x,Q(x)))$; E étant l'espérance mathématique ; donc :

$$D = E(d(X, Q(X)))$$

Cette espérance mathématique, quand elle existe, peut être décomposée de la manière suivante :

$$D = \sum_{i=1}^{k} p(x \in C_i) E[d(X, y_i) / x \in C_i]$$

x est une réalisation de *X* et y_i un élément de prototype *Y*.

$p(x \in C_i)$ est la probabilité discrète pour que *x* soit dans la classe C_i.

on obtient donc :

$$D = \sum_{i=1}^{k} p(x \in C_i) \int_{x \in C_i} d(x, y_i)\, p(x) dx$$

La minimisation de *D* nécessite, donc la minimisation de $\int_{x \in C_i} d(x, y_i)\, p(x) dx$

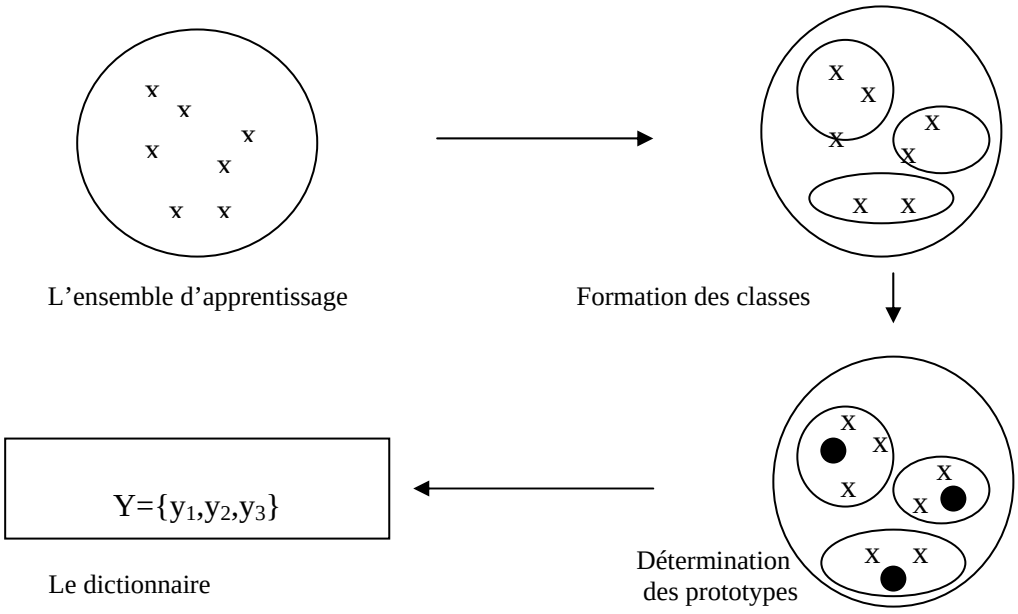


FIG 4.6 . ***PRINCIPE GENERAL DE LA QUANTIFICATION VECTORIELLE***

La Figure FIG 4.6 montre le principe générale de l'algorithme de la quantification vectorielle. A partir d'un ensemble d'apprentissage on cherche à déterminer des classes homogènes, ensuite on calcule pour chaque classe, un prototype représentatif. Ce processus est itératif jusqu'à d'un minimum global. Enfin, un dictionnaire est formé est contient donc l'ensembles des prototypes représentant les classes obtenues.

4.5 Programmation dynamique

4.5.1 Principe général

La programmation dynamique est une méthode de recherche d'optimum fondée sur le principe d'optimalité locale de Bellman : « Dans une séquence optimale de décisions, quelle que soit la première décision prise, les décisions subséquentes forment une sous-séquence optimale, compte tenu des résultats de la première décision ». Ce principe indique que tout chemin optimal est constitué de portions de chemins elles-mêmes optimales. En effet, si on considère un chemin optimal reliant A à B et un point quelconque M de ce chemin, les chemins de A à M et de M à B sont tous deux optimaux**[54]**.

4.5.2 Distance entre chaînes

La distance entre chaînes peut être calculée par minimisation de distances locales entre les composantes des chaînes et optimisation donc de la ressemblance.

Si l'on prend l'exemple de deux chaînes de caractères {abc} et {abde}, on s'aperçoit que la comparaison peut se faire en terme de transformations pouvant faire passer d'une chaîne à l'autre. Par exemple, pour passer de la première chaîne à la deuxième, il faut conserver les deux premiers caractères « ab », changer le troisième, remplacer la lettre « c » par la lettre « d » et ajouter un « e ». on remarque que ces opérations peuvent être effectuées à l'aide des trois transformations suivantes : SUB, DES, définies par :

SUBstitution	$x_i \rightarrow y_j$	$SUB(x_i,y_j)$
DEStruction	$x_i \rightarrow \lambda$	$DES(xi, \lambda)$
CREation	$\lambda \rightarrow y_j$	$CRE(\lambda,y_j)$

Où λ est le caractère vide. A chacune de ces transformations est associé un coût $c(x_i, y_j)$ pour SUB, $c(xi, \lambda)$ pour DES et $c(\lambda, y_j)$ pour CRE. La distance entre deux chaînes est égale au coût total des transformations les moins coûteuses de passage d'une chaîne à l'autre. Elle est appelée distance d'édition D(X, Y). Le schéma de transformation, appelé trace, reproduit par des flèches les transformations concernées.

Soit $X = \{x_1, x_2,x_3,x_4,x_5\}$ *et* $Y=\{y_1,y_2,y_3,y_4 ,y_5,y_6\}$.

Supposons que la trace de la transformation de *X* en *Y* soit la suivante :

$X : x_1 \; x_2 \; x_3 \; x_4 \; x_5$
$\quad\;\; \downarrow \; \downarrow \qquad \downarrow$
$Y : y_1 \; y_2 \; y_3 \; y_4 \; y_5 \; y_6$

Elle peut aussi s'écrire comme suit : $SUB(x_1, y_1)$; $SUB(x_2,y_2)$; $DES(x_3, \lambda)$; $SUB(x_4, y_3)$; $SUB(x_5,y_4)$, $CRE(\lambda,y_5)$; $CRE(\lambda, y_6)$. Chaque opération « trace » est affectée d'un coût et la distance est associée au coût minimal.

De manière générale, soient $X(n) = \{x_1, x_2, \ldots, x_n\}$ et $Y(m) =\{y_1, y_2, \ldots y_m\}$, deux chaînes de longueurs respectives n et m. il existe trois manières de progresser dans les deux chaînes pour aboutir à (x_r, y_k) :

Venir de (x_{r-1}, y_{k-1}) et faire $SUB(x_r, y_k)$;

Venir de (x_r, y_{k-1}) et faire $CRE(\lambda, y_k)$;

Venir de (x_{r-1}, y_k) et faire $DES(x_r, \lambda)$.

On choisit le chemin qui minimise la distance *D(r,k)* entre les deux sous-chaînes *X(r)* ($r \leq n$) et *Y*(*k*) ($k \leq m$) selon la formule suivante :

$$D(\mathrm{r,k}) = min \begin{cases} D(r-1,k-1) + c(x, y_k) \\ D(r,k-1) + c(\lambda, y_k) \\ D(r-1,k) + c(x_r, \lambda) \end{cases}$$

debut

D(0,0) = 0

Pour *r* de 1 à *n* faire

D(*r*,0) = D(*r*-1,0)+*c*(x_r,λ)

fpour

Pour *k* de 1 à *m* faire

D(0,*k*) = D(0,*k*-1)+*c*(λ,y_k)

fpour

Pour *r* de 1 à *n* faire

Pour *k* de 1 à *m* faire

d*1*=*D(*r-*1*,k-*1)*+c*(*x_r,y_k*)*

d*2*=*D(*r-*1*, k*)*+c*(*xr,*λ)*

d*3*=*D(*r, k-*1)*+c*(*λ*, y_k*)*

D*(*r,k*) = min(*d_1, d_2, d_3*)*

fpour

fpour

D(n,m)=distance entre les deux chaînes

fin

4.6 Application : codage et reconnaissance des sessions de navigations

Pour effectuer une typologie de sessions qui nous permet de comprendre le trafic dans le site, nous élaborons le processus suivant :

- Segmentation des sessions et codage des Urls selon §4.2.1 : incluant des Urls « Achat » ou « Non – Achat », les sessions peuvent être divisées selon les deux grandes catégories. Elles sont aussi représentées par des successions de neurones au lieu des Urls. En effet chaque ensemble d'Urls possède un représentant (prototype) dans la carte de Kohonen.
- Codage des sessions avec l'algorithme décrit dans §4.3 : cet algorithme que nous avons proposé, sert à transformer les sessions qui sont de longueurs différentes en vecteurs de même taille et de nature numérique.
- Application de l'algorithme LBG **[53]** : grâce à cet algorithme d'apprentissage, nous extrayons un dictionnaire. L'effectif de ce dernier sera de l'ordre d'une puissance de 2 vue la démarche binaire de l'algorithme en question.
- Extraction des autoroutes (prototypes de classes de sessions) : une fois l'apprentissage fini, nous utilisons le calcul de distorsion entre les sessions (représentées par leurs vecteurs) et les prototypes résultants de l'algorithme LBG. Nous avons donc, un ensemble de classes de sessions où chaque classe est représentée par le vecteur moyen des vecteurs – code des sessions qui appartiennent à cette classe. Contrairement au principe que nous avons défini dans l'algorithme Pdp, nous trions les variables de chaque vecteur moyen par ordre croissant sur la partie qui définit le poids de ces variables. Ce tri nous permet de retrouver l'ordre de passage du profil moyen de la classe dans la cartographie. Et par conséquent, nous pouvons construire les autoroutes de tous les classes issues de l'algorithme LBG.

14 classes de sessions ont été trouvées. Une classe de sessions est appelée autoroute dans ce qui suit.

N° Autoroute	Achat	Numéros de neurones de la séquence
1	Non	8 17 60 40 56 41 50 57 4 54 27 31 36 42 32 62
2	Oui	8 56 36 17 41 57 50 40 9 54 4 0 62
3	Non	8 9 4 62
4	Oui	8 31 41 51 56 4 5 9 0 54 57 17 1 62
5	Non	8 17 57 41 5 9 40 4 32 62
6	Oui	8 17 56 40 57 27 41 50 9 54 32 4 1 2 62
7	Non	8 17 56 5 4 42 41 54 32 62
8	Non	8 17 57 54 62
9	Oui	8 17 57 9 54 0 4 62
10	Non	8 60 54 62

11	Oui	8 27 50 42 31 5 40 9 41 56 32 4 57 0 54 2 17 1 62
12	Non	8 17 57 62
13	Oui	8 17 57 9 54 1 2 4 62
14	Non	8 17 62

TAB 4.1. ***LES DIFFERENTES AUTOROUTES EXTRAITES DU FICHIER LOG***

Les neurones 8 et 62 sont respectivement l'entrée et la sortie d'une session. D'autre part, en s'intéressant particulièrement aux comportements d'achat des internautes, nous avons ajouté une colonne indiquant si l'autoroute inclut un achat ou non. Ceci est déterminé par le fait qu'elle inclut ou non un des neurones : 0,1 ou 2 (correspondant respectivement aux Urls : « Souscription réserve », « Souscription perso », « Souscription auto »).

Nous pouvons maintenant visualiser clairement les autoroutes du tableau ci-dessus sur la cartographie du FIG 3.18.

FIG 4.7. ***REPRESENTATION DE QUELQUES AUTOROUTES SUR LA CARTOGRAPHIE DU SITE***

4.6.1 Reconnaissance de sessions par programmation dynamique : Résultats et comparaison

Nous allons essayer de valider l'algorithme de reconnaissance par DTW sur les 37174 sessions du site de 123Credit.com. Pour ceci, le seul critère de mesure que nous avions était le comportement Achat et

Non Achat : c'est à dire la vérification que, si une session était « Achat » (dont incluait un des neurones 0,1 ou 2) alors l'autoroute à laquelle le programme l'attribuait était elle – même « Achat » et vice-versa (une session « Non- Achat » était bien attribuée à une autoroute « Non – Achat » , i.e. qui ne contenait aucun des neurones 0, 1 ou 2. Nous avons donc mesuré les taux de reconnaissance obtenus selon ce critère pour l'algorithme de DTW d'une part et le calcul de distorsion (après codage par « poids de position , d écrit dans §4.3) et avons obtenu les résultats suivants :

	Sessions non achat	Sessions achat	Total
QV			
Autoroute non achat	99%	26%	35915
Autoroute achat	1%	74%	1259
DTW			
Autoroute non achat	99%	22%	35651
Autoroute achat	1%	78%	1523
TOTAL	35510	1664	37174

TAB 4.2. ***COMPARAISON ENTRE QV ET DTW SUR LA RECONNAISSANCE DES SESSIONS (ACHAT ET NON-ACHAT)***

On peut remarquer dans le tableau ci-dessus, que 99% des sessions NAC ont été attribuées par calcul de distorsion à des autoroutes NAC, 26% des sessions Achat à des autoroutes NAC etc.

Nous pouvons donner le tableau décrivant le détail sur la reconnaissance de chaque autoroute (TAB 4.3)

	Sessions non achat	Sessions achat	Total
QV			
Non achat (1)	96%	4%	192
Non achat (10)	63%	37%	672
Achat (11)	6%	94%	48
Non achat (12)	99%	1%	1089
Achat (13)	5%	95%	444
Non achat (14)	100%	0%	30721
Achat (2)	0%	100%	177
Non achat (3)	67%	33%	55
Achat (4)	0%	100%	6
Non achat (5)	94%	6%	342
Achat (6)	3%	97%	178
Non achat (7)	98%	2%	598
Non achat (8)	96%	4%	2246
Achat (9)	0%	100%	406
DTW			
Non achat (1)	81%	19%	155
Non achat (10)	97%	3%	275
Achat (11)	2%	98%	102
Non achat (12)	99%	1%	2252
Achat (13)	4%	96%	502
Non achat (14)	100%	0%	30334
Achat (2)	31%	69%	72
Non achat (3)	23%	77%	311
Achat (4)	27%	73%	22
Non achat (5)	88%	12%	131

Achat (6)	14%	86%	187
Non achat (7)	95%	5%	170
Non achat (8)	97%	3%	2023
Achat (9)	25%	75%	638
TOTAL	35510	1664	37174

TAB 4.3. ***COMPARAISON ENTRE QV ET DTW SUR LA RECONNAISSANCE DES SESSIONS (ACHAT ET NON-ACHAT) : DETAIL PAR AUTOROUTE***

4.6.2 Données de validation

Concernant les données à utiliser, nous avons été confronté au problème de données préalablement étiquetées qui nous auraient permis de mesurer les taux de succès (ou d'erreur) des algorithmes. En effet, Nous ne disposions que d'un fichier Log « brut » dont les sessions de navigations n'étaient ni reconnues comme étant semblables à tel « représentant », ni regroupées en classes. Afin de pallier ce problème, nous avons développé un générateur de sessions artificielles.

4.6.2.1 Génération de sessions artificielles étiquetées

Ce générateur fonctionne de la manière suivante : on lui fournit un certain nombre de représentants – ou prototypes « ou autoroutes » - ainsi que des paramètres de modification de ces autoroutes pour la génération des sessions. Le générateur crée alors de manière semi – aléatoire (à la fois en utilisant les paramètres précédents et en introduisant une part d'aléatoire) un certain nombre – également fourni en entrée par l'utilisateur – de sessions artificielles par autoroute. Une session étant obtenue à partir de la modification partielle d'un représentant, on sait d'avance qu'elle est théoriquement plus similaire à ce représentant qu'à tous les autres, et donc qu'elle est étiquetée par lui.

Plus précisément, les cinq types de modification que peut subir un représentant sont les suivants :

- Suppression d'un de ses neurones
- Remplacement d'un de ses neurones par un neurone voisin sur la carte
- Remplacement d'un de ses neurones par un neurone éloigné sur la carte
- Ajout à la suite d'un de ses neurones (i.e. à sa droite dans la séquence) d'un autre qui lui est voisin sur la carte
- Ajout à la suite d'un de ses neurones d'un autre qui lui est éloigné sur la carte.

Le nombre de modifications à effectuer par représentant est entré par l'utilisateur sous la forme d'un pourcentage du représentant à modifier. Ce taux ne pouvant dépasser 40% (pour que la session générée conserve un minimum de ressemblance avec son représentant). Par exemple, si une autoroute a une longueur de 10 neurones et qu'on fixe ce taux à 35%, la session aura 7 neurones en commun avec son représentant (le nombre de neurones modifiés est arrondi à l'entier inférieur le plus proche, soit ici : $E(10\times35/100)=E(3.5)=3$). Le programme effectue alors une boucle sur le nombre précédent au cours de laquelle, la position dans la séquence où une modification va être faite est déterminée aléatoirement et où le type de la modification est déterminée de manière semi-aléatoire. En effet, le programme demande à l'utilisateur de définir également parmi le nombre total de modifications à effectuer, quel sera le pourcentage de suppressions, de remplacements par un neurone voisin, de remplacements par un neurone éloigné, d'insertion d'un neurone voisin et d'insertion d'un neurone éloigné du neurone du représentant dont la position a été tirée et en fonction de sa valeur et des taux précédents, le type de modification à effectuer est choisi par le programme.

L'utilisateur fournit aussi au programme un pourcentage de sessions qui seront strictement identiques à l'autoroute (ce qui est susceptible d'arriver dans un fichier Log réel). Là aussi, en fonction de cette valeur et d'un troisième nombre aléatoire entre 1 et 100, le générateur décide de modifier le représentant (de la manière décrite ci-dessus) ou non.

Au final, le générateur renvoie donc en sortie trois fichiers texte :

Dans le premier qui est utilisé en entrée du programme de classification, sont stockées toutes les sessions générées selon le format suivant (une ligne par séquence) :

« No de session 1^{er} Neurone $2^{ème}$ Neurone … Neurone de sortie »

Le deuxième, qui est traité par le programme de reconnaissance contient toutes les sessions générées au format « DTW », c'est à dire qu'une séquence « Ni Nj Nk » par exemple sera représentée par un identifiant (numéro) de session sur la première ligne, puis (sur les lignes suivantes) par une matrice à 3 colonnes et L lignes, les éléments des colonnes étant les L composantes, sur une carte topologique qui sera décrite ci-dessous, des neurones Ni, Nj et Nk respectivement.

Enfin, le troisième fichier est un échantillon dont la taille, c'est à dire le nombre de sessions, est fixée par l'utilisateur de sessions extraites parmi toutes les sessions générées ci-dessus, au format DTW.

4.6.3 Résultats et comparaison avec le calcul de distorsion pour la reconnaissance de prototypes sur les données artificielles

Après avoir développé le générateur de sessions artificielles, nous l'avons fait fonctionner à partir des 14 autoroutes précédentes, en lui demandant de créer 100 sessions par autoroute (soit 1400 sessions en tout), et avec différentes valeurs pour les paramètres de modification des autoroutes. Nous avons alors mesuré le taux de reconnaissance exact des 1400 sessions, i.e. le nombre de fois qu'une session est reconnue par les procédures DTW et calcul de distorsion) à l'autoroute à partir de laquelle elle avait été générée (cette information nous a été donnée par l'identifiant de la session, puisque les 100 premières sessions avaient été générées à partir de la 1ère autoroute, les 100 suivantes à partir de la 2ème etc). Le tableau suivant récapitule les résultats obtenus en fonction des paramètres de génération et des algorithmes utilisés.

Nous avons défini les paramètres du générateur de sessions artificielles comme suit :

%MA	%SN	%RNV	%RNE	%INV	%INE	TR par QV	TR par DTW
35	32	17	17	17	17	69	95
20	32	17	17	17	17	84	99
40	60	10	10	10	10	60	92
40	10	60	60	10	10	64	98
40	10	10	10	60	10	67	96
40	10	10	10	10	10	76	98
40	10	10	10	10	60	76	98

TAB 4.4 ***PARAMETRES D'ENTREE POUR LE GENERATEUR DE SESSIONS ARTIFICIELLES ET TAUX DE RECONNAISSANCE PAR QV VS DTW***

MA : modification d'autoroute

SN : suppression de neurones

RNV : remplacement par neurones voisins

RNE : remplacement par neurones éloignés

INV : insertion par neurones voisins

INE : insertion par neurones éloignés

TR : taux de reconnaissance

Ceci donne un taux de reconnaissance de 70,86% pour le calcul de distorsion (QV) et 96,57% pour la DTW. De plus, nous avons calculé les matrices de confusion associés à la reconnaissance des différentes autoroutes (indiquant par autoroute, le nombre d'attributions à toutes les autoroutes des sessions générées à partir de l'autoroute en question) pour le premier jeu de données et avons obtenu les résultats suivants :

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14
A1	96	1	0	0	1	1	0	0	0	0	0	0	1	0
A2	11	42	0	0	3	12	0	0	29	0	0	0	3	0
A3	0	0	100	0	0	0	0	0	0	0	0	0	0	0
A4	1	10	0	0	1	35	0	0	13	0	10	0	30	0
A5	4	0	17	0	65	0	5	1	3	0	0	0	5	0
A6	8	2	0	0	0	83	0	0	0	0	0	0	7	0
A7	45	0	0	0	0	0	47	1	1	0	0	0	6	0
A8	0	0	0	0	0	0	19	56	0	0	0	25	0	0
A9	0	0	6	0	11	0	0	8	67	0	0	0	8	0
A10	0	0	0	0	0	0	0	0	0	100	0	0	0	0
A11	2	11	0	0	0	23	0	0	0	0	64	0	0	0
A12	0	0	0	0	1	0	0	5	0	0	0	69	0	25
A13	0	0	2	0	3	5	0	6	1	0	0	0	83	0
A14	0	0	0	0	0	0	0	0	0	0	0	0	0	100

TAB 4.5. ***MATRICE DE CONFUSION POUR LA RECONNAISSANCE PAR QV***

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14
A1	100	0	0	0	0	0	0	0	0	0	0	0	0	0
A2	0	100	0	0	0	0	0	0	0	0	0	0	0	0
A3	0	0	100	0	0	0	0	0	0	0	0	0	0	0
A4	0	0	0	100	0	0	0	0	0	0	0	0	0	0
A5	0	0	0	0	99	0	0	0	1	0	0	0	0	0
A6	0	0	0	0	0	100	0	0	0	0	0	0	0	0
A7	0	0	0	0	0	0	100	0	0	0	0	0	0	0
A8	0	0	0	0	0	0	0	71	0	12	0	17	0	0
A9	0	0	1	0	0	0	0	4	88	0	0	0	7	0
A10	0	0	0	0	0	0	0	0	0	85	0	0	0	15
A11	0	0	0	0	0	0	0	0	0	0	100	0	0	0
A12	0	0	0	0	0	0	0	0	0	0	0	88	0	12
A13	0	0	0	0	0	0	0	0	1	0	0	0	99	0
A14	0	0	0	0	0	0	0	0	0	0	0	0	0	100

TAB 4.6. ***MATRICE DE CONFUSION POUR LA RECONNAISSANCE PAR DTW***

Ces matrices se lisent de la manière suivante : par exemple, on voit sur la première que sur les 100 sessions générées à partir de l'autoroute 9, 6 ont été reconnues comme étiquetée par l'autoroute 3, 11 par l'autoroute 5, 8 par l'autoroute 8, 67 par l'autoroute 9 (la bonne) et 8 par l'autoroute 13 etc.

Il faut préciser que la méthode de profiling que nous avons proposée dans ce chapitre cause un problème de perte d'information. Ce problème est dû au passage de la nature symbolique des sessions vers les vecteurs issus d'un codage numérique. Dans ce qui suit, nous allons procéder par la modélisation par mélange de chaînes de Markov dont les états seront représentés par les neurones de la carte topologique que nous avons développée pour la modélisation comportementale.

La méthode que nous allons développer, fera l'objet d'un processus automatique qui assurera une analyse descriptive des sessions tout en gardant leur nature symbolique bien qu'elle soit floue et difficile à comprendre.

4.7 Conclusion

Élaborant un processus d'analyse de séquences basé sur une nouvelle méthode de codage et une quantification vectorielle, nous avons pu résumer l'ensemble des sessions dans des catégories représentant des profils significatifs prêts à être représentés sur une cartographie synthétique.

La programmation dynamique a été utilisée pour déterminer la ressemblance entre les sessions de navigation du site (de longueurs différentes) et ainsi permettre de reconnaître de quel comportement de navigation « type » une session quelconque est la plus proche. Les sessions traitées se présentaient sous la forme d'une séquence de « neurone » de la cartographie du site.

Ce travail nous a permis d'avoir une première compréhension des internautes dont les comportements sont préalablement assez difficiles à cerner .

Chapitre V

5 Modèles de mélanges : Une autre approche pour le profiling

Plusieurs exemples de classification à base de mélanges normaux ont fait l'objet de plusieurs travaux. Ce chapitre traite le concept des modèles de mélanges, plus particulièrement avec des mélanges de distributions discrètes qui ont toujours été exploités comme des techniques de classification automatique.

5.1 Introduction

Les mélanges de distributions avaient particulièrement été utilisés comme étant des modèles de manière extensive, dans une large variété de situations pratiques **[56]**. Des situations où les données peuvent être vues comme deux ou plusieurs populations mixées dans des proportions variables. Par exemple, la densité d'une observation est prise pour être un mélange de deux densités normales univariées avec le même sens, mais seulement dans le cas où la seconde composante a la plus grande variance. Cette famille avait été introduite pour modéliser une population qui suit une distribution normale à l'exception des rares occasions où des observations atypiques sont grossièrement enregistrées.

5.2 Une approche de mélange de vraisemblance pour la classification

Dans cette section, nous formulons une estimation de vraisemblance pour les modèles de mélanges finis. Ensuite nous définissons l'approche de mélange de vraisemblance pour la classification pour des ensembles de données de la forme de vecteurs. Les observations multivariées dans un ensemble de *n* entités, peuvent être représentées comme suit : x_1, x_2,…,x_n, où chaque x_j est un vecteur de *p* dimensions. Chaque x_j peut être vu comme une superpopulation *G* dans quelques proportions π_1, …, π_g, respectivement, tel que :

$$\sum_{i=1}^{g} \pi i = 1 \; et \;\; \pi i \geq 0 \qquad (\text{i} = 1, . ..,g)$$

La fonction de densité (f.d.) d'une observation *x* dans *G*, peut donc être représentée sous forme de mélange fini :

$$f(x\,;\,\Phi) = \sum_{i=1}^{g} \pi f_i(x;\theta) \qquad (1)$$

Où : $f_i(x, \Phi)$ est la f.d. correspondant à G_i , et θ dénote le vecteur de tous les paramètres inconnus associés aux formes paramétriques adoptées pour ces *g* densités. Par exemple, dans un cas particulier de densités normales multivariées , θ représente le vecteur moyenne μ_i et les éléments distincts des matrices de covariance $\sum_i$ pour i=1,…,g, les vecteurs $\Phi=(\pi',\theta')'$ de tous les paramètres inconnus appartiennent à un espace de paramètre Ω .

Strictement parlant, puisque la somme des éléments de π est égale à 1, alors une des proportions de mélange π_i est redondante. Mais, nous ne pouvons pas modifier Φ explicitement. Cependant, il pourra être implicitement supposé que un des π_i soit supprimé de Φ. Dans toute cette étude nous pouvons soumettre à *Gi* n'importe quelle population ou groupe sans se soucier de la situation sous-jacente. Bien que l'utilisation du premier terme est réellement la plus appropriée, pour le cas où la superpopulation est a priori composée de *g* composantes distinctes, par rapport au cas usuel dans la classification où une partition artificielle est imposée sur les données.

Dans la formule (1), les fonctions de densité sont en accord avec une mesure arbitraire dans R^p. Dans ce cas, $f_i(x ; \theta)$ peut être une fonction de masse par l'adoption d'une mesure de calcul. Dans plusieurs applications, les densités constituantes $f_i(x ; \theta)$ sont prises dans la même famille paramétrique. Nous prenons $F(x ; \theta)$ comme la fonction de distribution correspondante à la densité de mélange de (1). Anderson(1979)**[55]** avait appelé $F(x ; \theta)$ la densité composée et avait réservé le mot « mélange » pour le distinguer du procédé d'échantillonnage séparé.

Il est supposé à présent, que $x_1,\ldots, x_n$ sont des valeurs observées d'un échantillon aléatoire de taille *n* à partir de G_i . Ce sont des valeurs réalisées par *n* variables aléatoires indépendamment et identiquement distribuées (iid) avec la fonction de distribution commune $F(x ; \theta)$. Cela peut être approprié dans plusieurs situations pratiques.

Notre hypothèse sur l'ensemble des données présent peut être écrite comme suit :

$$x_1,\ldots, xn \overset{iid}{\approx} F(x ; \Phi), \Phi \in \Omega. \qquad (2)$$

La fonction de vraisemblance pour Φ peut être facilement formée sous (2) et une estimation de Φ peut être obtenue comme une solution de l'équation de vraisemblance suivante :

$$\partial L(\Phi)/\partial\Phi = 0,$$

où $L(\Phi)$ représente le log de vraisemblance. Nous appellerons notre solution Φ' pour l'équation de vraisemblance. Malheureusement, avec les modèles de mélange, l'équation de vraisemblance possède plusieurs racines dont la résolution et le choix de la racine appropriée à Φ' ne sont pas évidents. Mais brièvement, l'objectif de l'estimation de vraisemblance est de déterminer Φ' pour chaque n . Alors, cela définit une séquence de racines pour l'équation de

vraisemblance qui est consistante et asymptotiquement efficace. En général, pour les modèles d' estimation la vraisemblance possède habituellement un maximum global dans l'espace des paramètres. Donc typiquement, une séquence de racines pour l'équation de vraisemblance avec des propriétés asymptotiques désirées est fournie par le fait de prendre Φ' pour chaque n, comme racine qui maximise globalement la vraisemblance. Par conséquent, Φ' est l'estimateur de vraisemblance maximum. Cependant, pour les modèles de mélange, les exemples peuvent être facilement trouvés quand la vraisemblance est illimitée. Alors l'estimateur de vraisemblance maximum n'existe pas.

Une fois Φ' est obtenue, les estimations des probabilités a posteriori des individus d'une population peut être formée pour chaque x_j pour donner une classification probabiliste. La probabilité a posteriori que x_j appartienne à G_i est donnée par :

$$\tau_i(x_j ; \Phi) = pr(j^{ème} \text{ entité} \in G_i / x_j ; \Phi)$$

$$= \pi_i f_i(x_j ; \theta) \quad / \quad \sum \pi_t f_t(x_j;\theta) \quad (i= 1, \ldots, g). \qquad (3)$$

Un partitionnement de x1, …, xn dans g classes non chevauchées peut être effectué par l'attribution de chaque x_j à la population dans laquelle il possède la plus grande probabilité d'appartenance a posteriori dans G_t si :

$$\tau_t(x_j ; \Phi') > \tau_i(x_j ; \Phi') \ (i = 1,.. ., g; i \neq t) \qquad (4)$$

Ultérieurement, nous notons respectivement $\tau_t(x_j ; \Phi')$ et $\tau_i(x_j ; \Phi')$ par τ_{tj} et τ_{ij} .

5.3 Identifiabilité

A fin d'être capable d'estimer tous les paramètres dans Φ à partir de $x_1,\ldots,xn$, il est nécessaire qu'ils soient identifiables. En général, la famille paramétrique des p.d.f f(x ; Φ) est dite identifiable si les valeurs distinctes de Φ déterminent les membres distincts de la famille. Cela peut être interprété comme dans le cas où f(x ; Φ) définit une classe de densités de mélange selon (1). Une classe de mélanges finis est dite identifiable pour $\Phi \in \Omega$ si pour n'importe quels deux membres :

$$f(x ; \Phi) = \sum_{i=1}^{g} \pi_i f_i (x; \theta),$$

Et

$$f(x\,;\,\Phi^*) = \sum_{i=1}^{g^*} \pi_i^{\,*}\, f_i\,(x;\,\theta^*),$$

alors

$$f(x;\,\theta) \equiv f(x;\,\theta^*)$$

si et seulement si g = g* et nous pouvons les étiquettes composantes , donc

$\pi_i = \pi_i^{\,*}$ et $f_i(x;\,\theta) \equiv f_i(x;\,\theta^*)$ (i = 1,. ..,g).

Ici, ≡ représente l'égalité des densités pour presque tous les x relatifs à la mesure sous-jacente dans R^p appropriée à f(x ; Φ).

5.4 Classification par modèles de mélange de distribution

5.4.1 Principe général

L'idée fondamentale de ce type de classification est que les données observées sont générées par un mélange de *K* distributions statistiques. Chaque distribution représente une classe et possède une certaine variabilité. Un tel modèle est appelé par les statisticiens, un « modèle de mélange à *K* composantes » **[58]**. Initialement, bien sûr, le modèle n'est pas connu ; on ne dispose que des observations. Mais on peut appliquer des techniques statistiques classiques aux données pour « apprendre » le modèle, c.à.d :

- le nombre de composantes K
- les probabilités d'appartenance *a priori* d'un individu aux classes, c'est à dire les « poids » associés aux composantes du modèle.
- les paramètres de chaque composante du modèle

Une fois le modèle appris, on peut calculer la probabilité d'appartenance de chaque individu aux différentes classes et en déduire une classification des individus.

5.4.2 Formalisation mathématique

Nous décrivons un modèle de mélange à K composantes de la manière suivante :

Soient **X** une variable aléatoire multivariée prenant des valeurs correspondantes aux observations et C une variable à valeurs

discrètes dans $\{c_1, \ldots, c_K\}$ représentant l'appartenance d'une observation à la classe c_i. Il est important de noter que pour un X donné, la valeur de C est inconnue : C est donc une variable « cachée » (non observée).

Alors, la probabilité pour une observation x d'être générée (ou, en d'autres termes, la probabilité que **X** vaille **x**) est un modèle de mélange, c'est à dire une combinaison linéaire des composantes :

$$p(X=x/\theta)=\sum_{k=1}^{K} p(C=c_k/\theta)p_k(X=x/C=c_k,\theta_k) = \sum_{k=1}^{K} \pi_k\, p_k(X=x/C=c_k,\theta_k) \qquad (5)$$

où

- $\pi_k = p(C = c_k / \theta)$ est la probabilité marginale (à priori) de la $k^{ème}$ classe, c'est à dire le « poids » de cette composante dans le modèle ; donc $\pi_k \geq 0$ et $\Sigma_k \pi_k = 1$
- $p_k(\mathbf{x} / \theta_k)$ est la distribution de **X** dans la $k^{ème}$ classe (qui décrit la distribution des attributs des séquences de cette classe
- θ_k sont les paramètres de p_k
- $\theta = \{\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K\}$ sont les paramètres du modèle de mélange

Soient $D = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ un ensemble d'observations supposées indépendamment et identiquement distribuées (i.i.d). Alors la probabilité – les statisticiens parlent de « vraisemblance » - qu'elles soient simultanément générées par le modèle de mélange M est :

$$V_M(\theta; D) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k\, p_k(\mathbf{x_i} / \theta_k)$$

En pratique, pour transformer le produit ci-dessus en somme, on considère généralement plutôt le logarithme de la vraisemblance :

$$L_M(\theta; D) = \log[V_M(\theta; D)]$$
$$= \log\left[\prod_{i=1}^{N}\sum_{k=1}^{K}\pi_k p_k(\mathbf{x_i}/\theta_k)\right]$$
$$= \sum_{i=1}^{N}\log\left[\sum_{k=1}^{K}\pi_k p_k(\mathbf{x_i}/\theta_k)\right] \qquad (6)$$

Le problème consiste donc à estimer les paramètres θ du modèle M qui maximisent le L_M de l'équation (6) ci-dessus :

$$\theta^{ML} = argmax_\theta L_M(\theta; D) \qquad (7)$$

les lettres « ML » étant les initiales de « Maximum Log-Likelihood ».

On cherche donc les valeurs de θ qui annulent toutes les dérivées partielles de L_M :

$$\frac{\partial L_M(\theta;D)}{\partial\theta}=0 \qquad (8)$$

Cette méthode de maximisation de la vraisemblance **[59]** est l'une des méthodes d'estimation les plus utilisées dans de nombreux domaines d'application des statistiques : traitement du signal et de l'image, communications, reconnaissance des formes, apprentissage connexionniste etc. Cependant, on ne peut pas toujours trouver de solutions analytiques aux équations *(8)* ci-dessus. Les méthodes d'optimisation numérique directe (Newton-Raphson, gradient etc.) de la vraisemblance permettent d'estimer de telles solutions, mais nécessitent souvent un travail préalable important et des temps de calculs élevés.

Pour une certaine famille de problèmes statistiques – les problèmes à « données incomplètes » - , une alternative au calcul numérique direct des θ^{ML} a été introduite en 1977 par Dempster et al.(voir **[59]**) : l'algorithme « Expectation-Maximization » ou « EM » qui fera l'objet du § 5.4.3 suivant.

5.4.3 Des distributions bien adaptées aux séquences : les chaînes de Markov

Les distributions p_k habituellement utilisées pour la classification à base de modèle de mélange sont les Gaussiennes. Celles-ci sont

souvent bien adaptées quand les données sont des valeurs numériques uniques mais sont inappropriées dans le cas (qui est le nôtre) où les observations sont des séquences de valeurs (ou d'évènements). Celles-ci sont en revanche « naturellement » représentées par des « chaînes de Markov », qui sont définies ci-dessous :

5.4.3.1 Définition générale

Soient une suite X_0, X_1, …, de variables aléatoires prenant des valeurs dans un ensemble commun {0, 1, …, M}. Ce modèle peut servir à représenter l'état d'un système au cours du temps, X_n désignant cet état au temps n. Dans ce cadre, on dit que le système se trouve dans l'état i au temps n si X_n = i. La suite des variables considérées est appelée **chaîne de Markov [56]**si à partir de tout état i la probabilité P_{ij} de passer immédiatement après à l'état j est constante au cours du temps (et ne dépend pas de ce qui s'est passé auparavant). Plus précisément, la condition requise – appelée la « propriété de Markov » - s'écrit, pour tout ensemble de réels i_0, i_1, …, i_{n-1}, i, j :

$$P\{ X_{n+1} = j / X_n = i, X_{n-1} = i_{n-1}, \ldots, X_1 = i_1, X_0 = i_0 \} = P\{ X_{n+1} = j / X_n = i\} = P_{ij}$$

Les grandeurs P_{ij} , où $0 \leq i \leq M$ et $0 \leq j \leq M$, sont appelées **probabilités de transition** de la chaîne de Markov et vérifient (par définition d'une probabilité) les propriétés:

$$P_{ij} \geq 0 \text{ et } \sum_{j=0}^{M} P_{ij} = 1, \text{ pour } i = 0,1,\ldots,M$$

La matrice T définie par $(T)_{ij} = P_{ij}$ est appelée **matrice de transition**.

La connaissance de la matrice de transition ainsi définie et de la distribution de X_0 permet théoriquement de calculer toutes les probabilités désirées. La probabilité conjointe de X_0, …, X_n par exemple est calculable ainsi :

$$\begin{aligned} P\{X_n = i_n, X_{n-1} = i_{n-1},\ldots,X_0 = i_0\} &= P\{X_n = i_n / X_{n-1} = i_{n-1},\ldots,X_0 = i_0\} P\{X_{n-1} = i_{n-1},\ldots,X_0 = i_0\} \\ &= P_{i_{n-1},i_n} P\{X_{n-1} = i_{n-1},\ldots,X_0 = i_0\} \end{aligned}$$

et la répétition de cet argument finit par montrer que la probabilité ci-dessus vaut :

$$= P_{i_{n-1},i_n} P_{i_{n-2},i_{n-1}} \ldots P_{i_1,i_2} P_{i_0,i_1} P\{X_0 = i_0\}$$

5.4.4 Estimation des paramètres du modèle : l'algorithme EM

Le but de l'algorithme « Expectation-Maximization » ou « EM » est de trouver les valeurs des paramètres θ (ici, les π_k,θ_k^I et θ_k^T) qui, étant donné un ensemble d'observations D, maximisent le logarithme de la vraisemblance $L_M(D,\theta)$ définie dans l'équation *(6)* ci-dessus. Après avoir été initialisées, les valeurs des θ sont mises à jour à chaque itération de l'algorithme de manière à faire augmenter la vraisemblance jusqu'à un maximum. Néanmoins, en général, EM ne peut trouver qu'un maximum local.

5.4.4.1 Principe général

Notons l'estimation courante des paramètres optimaux à l'itération t θ^t et renotons, pour simplifier, les observations **X** et le logarithme de la vraisemblance ci-dessus L(θ). Examinons ce que devient $L(\theta^t)$ quand l'algorithme calcule une nouvelle valeur de θ :

$$L(\theta) - L(\theta^t) = \log P(\mathbf{X}/\theta) - \log P(\mathbf{X}/\theta^t) = \log \frac{P(\mathbf{X}/\theta)}{P(\mathbf{X}/\theta^t)}$$

En fonction de la valeur que l'on choisit pour θ, la valeur de L pourrait augmenter ou diminuer. On souhaiterait choisir θ de manière à maximiser le membre de droite de l'équation ci-dessus. Or en général, ceci ne peut être fait ; l'idée fondamentale sur laquelle repose l'algorithme EM est d'introduire des variables non observées (cachées) **C** , adaptées à la famille de modèles considérés, telles que si les **C** étaient connues la valeur optimale des θ pourrait être calculée facilement. Mathématiquement, on introduit **C** dans l'équation de la manière suivante :

$$\begin{aligned} L(\theta) - L(\theta^t) &= \log \frac{\sum_C P(\mathbf{X/C},\theta) P(\mathbf{C}/\theta)}{P(\mathbf{X}/\theta^t)} \\ &= \log \frac{\sum_C P(\mathbf{X/C},\theta) P(\mathbf{C}/\theta)}{P(\mathbf{X}/\theta^t)} \frac{P(\mathbf{C/X},\theta^t)}{P(\mathbf{C/X},\theta^t)} \end{aligned}$$

On peut alors appliquer l'inégalité de Jensen ($\log \Sigma_j \lambda_j y_j \geq \Sigma_j \lambda_j \log y_j$ si $\Sigma_j \lambda_j = 1$) et obtenir :

$$L(\theta) - L(\theta^t) \geq \sum_C P(\mathbf{C/X},\theta^t) \log \frac{P(\mathbf{X/C},\theta) P(\mathbf{C},\theta)}{P(\mathbf{X}/\theta^t) P(\mathbf{C/X},\theta^t)}$$

Ceci peut être réécrit comme $L(\theta) \geq L(\theta^t) + \Delta(\theta,\theta^t)$, où

$$\Delta(\theta / \theta_i) = \sum_C P(\mathbf{C/X},\theta^t) \log \frac{P(\mathbf{X/C},\theta)P(\mathbf{C},\theta)}{P(\mathbf{X}/\theta^t)P(\mathbf{C/X},\theta^t)}$$

L'idée est alors de trouver la valeur des θ qui maximisent $L(\theta^t)$ + $\Delta(\theta,\theta^t)$ (on espère que grâce à l'inclusion des variables cachées, cela sera plus facile que le problème initial de maximisation de L) :

$$\theta^{t+1} = \text{argmax}_\theta \, L(\theta^t) + \sum_C P(\mathbf{C/X},\theta^t) \log \frac{P(\mathbf{X/C},\theta)P(\mathbf{C},\theta)}{P(\mathbf{X}/\theta^t)P(\mathbf{C/X},\theta^t)}$$
$$= \text{argmax}_\theta \sum_C P(\mathbf{C/X},\theta^t) \log\left[P(\mathbf{X/C},\theta)P(\mathbf{C},\theta)\right] = \text{argmax}_\theta \, E_{Z/X,\theta^t}\left[\log P(\mathbf{X},\mathbf{C}/\theta)\right]$$

,où E est l'espérance mathématique ou « Expectation » en anglais.

Cette équation résume à elle seule l'algorithme EM : l'étape « E » est le calcul de l'espérance - qui est la valeur attendue des variables cachées (**C**) compte tenu des observations (**X**)et des valeurs courantes θ^t des paramètres θ.– et l'étape « M » est la maximisation (de L grâce aux nouvelles valeurs θ^{t+1} des θ).

5.4.4.2 Importance de l'initialisation

On démontre (voir par exemple **[57]**) que cet algorithme converge vers un maximum **local** de la vraisemblance. En conséquence, la phase d'initialisation de l'algorithme est très importante, car une mauvaise initialisation pourrait aboutir à un mauvais modèle ; alors que, a contrario, une bonne initialisation entraînera une convergence vers le maximum global de la vraisemblance.

C'est pourquoi une des contributions les plus importantes de ce travail est la proposition d'une nouvelle méthode d'initialisation, qui nous semble beaucoup plus pertinente que celle utilisée par **[58]**.

5.4.4.3 Limites de l'algorithme EM

L'algorithme EM appliqué à la classification a quelques limites. D'abord, la convergence peut être très lente, bien qu'en pratique cela ne se produise pas si les classes sont bien séparées et si l'initialisation est bonne. Ensuite, le calcul des $\hat{e}_{ik}$ doit être effectué pour toutes les valeurs de i et de k, ce qui peut demander un temps de calcul important si le nombre d'observation et/ou de classes est très grand. Enfin, l'algorithme peut échouer si il y a très peu d'observations dans les classes ou si le nombre réel de classes est inférieur au nombre de composantes du modèle.

5.4.5 Détermination du nombre de classes optimal

5.4.5.1 Problématique et solution classique

L'algorithme ci-dessus fonctionne en général bien mais présente une limitation majeure : il impose de fixer a priori le nombre de classes K qui vont être obtenues alors que, en pratique, c'est un paramètre dont on dispose rarement. On voudrait donc pouvoir trouver, également de manière automatique, le nombre de classes « optimal » (c'est à dire « naturel ») du jeu de données.

La solution classiquement utilisée en statistique pour résoudre ce problème est d'introduire un critère (numérique) qui mesure l'adéquation du modèle (et donc de son nombre de composantes) au données traitées. On calcule alors ce critère pour différentes valeurs de K et celle qui le maximise est considérée comme étant le nombre de classes optimal (et la classification associée est considérée comme la meilleure).

5.4.5.2 Exemples de critères à optimiser

De nombreux critères ont été proposées dans la littérature :

- le plus ancien est le critère d'information d'Akaike **[61]**, « Akaike Information Criterion » ou « AIC », qui est défini par :

$$AIC(M) = -2\log(L_M) + 2\nu_M,$$

où ν_M est le nombre de paramètres indépendants du modèle M, et L_M est la vraisemblance définie plus haut.

- le plus célèbre est le critère d'information de Bayes, « Bayesian Information Criterion » ou « BIC », défini par :

$$BIC(M) = -2\log(L_M) + \nu_M \log(N),$$

où N est le nombre total d'observations.

- Cadez et al. **[58]** ont utilisé un score reflétant le nombre moyen de bits nécessaires pour coder la demande de visualisation d'une catégorie de pages (qui correspond, dans notre cas, à un neurone) par l'utilisateur, et défini par :

$$score(K) = \frac{\sum_{j=1}^{N} \log_2 p(\mathbf{X} = \mathbf{x_j} / \theta^K)}{\sum_{i=1}^{N} longueur(\mathbf{x_i})},$$

où longueur($\mathbf{x_i}$) est la longueur de la $i^{ème}$ séquence.

5.5 Application : Modélisation des sessions de navigation Web

Le fait qu'une session de navigation sur le Web vérifie effectivement la propriété de Markov n'est pas évident : en effet, si par exemple quelqu'un est allé d'une page *i* à une page *j* à un instant *t* au cours d'une session, il n'est pas sûr que cette transition ait la même probabilité d'occurrence à un instant t + t' quelconque. Néanmoins, sans preuve du contraire, nous supposons que ceci est vrai en première approximation et, compte tenu des avantages de ce modèle (en particulier, sa relative simplicité), nous avons choisi de l'utiliser pour notre application.

Pour ceci, on utilise simplement la correspondance suivante avec les notations ci-dessus : n est le $n^{ème}$ « clic » de la session, et i est le numéro du neurone visité (un neurone étant une classe de pages) au cours de ce clic (en d'autres termes, X_n = i veut dire « lors du $n^{ème}$ clic, le $i^{ème}$ neurone est visité »).

D'autre part, la variable aléatoire **X** introduite auparavant, est alors une séquence de longueur arbitraire de variables décrivant le chemin de navigation dans le site : **X** = (X_1, …, X_L). La variable X_i prend une valeur x_i indiquant le numéro du neurone vu par un utilisateur, et la séquence (x_1, x_2, …,x_L) indique donc que l'utilisateur voit d'abord le neurone x_1, puis le neurone x_2 etc. Il est important de noter que le dernier neurone vu (x_L) sera toujours le neurone de sortie, indiquant ainsi un état de « fin » de la navigation.

Compte tenu de ces notations, la distribution p_k s'écrit alors :

$$p_k(\mathrm{x}/\theta_k) = p_k(\mathrm{X}=\mathrm{x}/C=c_k,\theta_k) = p(x_1/\theta_k^l)\prod_{i=2}^{L} p(x_i/x_{i-1},\theta_k^T) \qquad (9)$$

où $\theta_k = \{\theta_k^I, \theta_k^T\}$, θ_k^I est la distribution de probabilité du neurone vu au cours du premier clic (correspondant au X_0 ci-dessus) dans la $k^{ème}$ classe (c'est donc un vecteur de dimension le nombre total de neurones N_N) et θ_k^T représente les probabilités de transition d'un neurone aux autres entre deux clics consécutifs, toujours pour la $k^{ème}$ classe (et c'est donc une matrice de dimension $N_N \times N_N$).

Chaque variable X_i est discrète, $p(x_1 / \theta_k^I)$ est une distribution multinômiale et $p(x_i/ x_{i-1}, \theta_k^T)$ est un ensemble de distributions multinômiales.

Ce modèle permet de prendre en compte la requête initiale d'un utilisateur, la dépendance entre deux requêtes consécutives, et –

grâce au neurone de sortie (ou à l'état de « fin ») – le dernier neurone visité.

Concrètement, dans notre cas, pour une observation $\mathbf{x_i}$ donnée, calculer l'espérance de C associée consiste à calculer ses probabilités d'appartenance à chacune des classes (ou composantes du modèle). Notons ainsi $\hat{e}_{ik}$ la probabilité d'appartenance de l'observation $\mathbf{x_i}$ à la $k^{ème}$ classe (ou composante du modèle). D'après la formule de Bayes généralisée, on a :

$$\hat{e}_{ik} = p(C=c_k / x_i, \theta) = \frac{\pi_k \, p_k(x_i/\theta_k)}{\sum_{j=1}^{K} \pi_j \, p_j(x_i/\theta_j)} \qquad (10)$$

où les $p_j(\mathbf{x_i}/\theta_j)$ sont obtenus à partir de l'équation *(9)* précédente.

L'utilisation de l'algorithme EM se fait donc de la manière suivante :

- Initialisation la plus pertinente possible (voir plus loin « Importance de l'initialisation ») des paramètres θ (ici, les π_k, θ_k^I et θ_k^T pour tous les k)
- **Itération** des deux étapes :
- Étape E : calcul des $\hat{e}_{ik}$ pour tous les i et tous les k grâce aux équations *(9)* et *(10)* ci-dessus appliquées aux valeurs θ^t des paramètres θ à l'itération (courante) t
- Étape M : calcul des nouvelles (à l'itération t+1) valeurs θ^{t+1} des paramètres θ à partir des valeurs de C ($\hat{e}_{ik}$) trouvées à l'étape E, en « faisant comme si » ces valeurs étaient des données (observations) réelles et connues.

jusqu'à une certaine stabilité (ou convergence) de la vraisemblance

En pratique, le calcul des θ^{t+1} à l'étape M peut se faire de deux façons, suivant la manière dont les observations sont attribuées aux classes :

- Attribution « douce » : dans ce cas, à l'étape E, chaque observation est attribuée à toutes les classes avec le poids $\hat{e}_{ik}$ et à l'étape M, le recalcul des paramètres associés à une classe donnée se fait à partir de toutes les observations (puisque toutes celles-ci sont attribuées à toutes les classes), mais la contribution de chacune est pondérée par son poids. Plus

précisément, ce calcul se fait de la manière suivante : pour les π_k, on additionne les $\hat{e}_{ik}$ pour tout i (et avec k fixé) ; pour les θ_k^I on somme, pour chaque neurone, les $\hat{e}_{ik}$ pour les valeurs de i dont les séquences correspondantes commencent par le neurone en question ; et pour les θ_k^T, en incrémentant les probabilités de transition d'un neurone a à un neurone b de $\hat{e}_{ik}$ à chaque fois que, au sein de la $i^{ème}$ séquence, les neurones a et b sont consécutifs. Enfin, après avoir effectué ceci pour tous les i (toutes les séquences), on normalise les π_k, θ_k^I et θ_k^T en divisant les valeurs obtenues respectivement par le nombre total de séquences, la somme des θ_k^I (sur tous les neurones, à k fixe) et la somme des θ_k^T sur une ligne de la matrice de transition, pour toutes les lignes (à k fixe). Cette normalisation permet de respecter les contraintes de somme égale à un des probabilités : $\Sigma_k \ \pi_k = 1$, $\Sigma_n \ \theta_k^I$ (neurone n) = 1 et $\Sigma_j \ P_{ij} = 1$ (où P_{ij} sont les probabilités de transition correspondantes aux θ_k^T).

Attribution « dure » : dans ce cas, à l'étape E, chaque observation est attribuée uniquement à la classe k pour laquelle $\hat{e}_{ik}$ est maximale (à i fixé) et à l'étape M, les paramètres d'une composante du modèle (classe) sont recalculés uniquement à partir des observations attribuées à la classe en question, chaque observation contribuant de manière égale (cad avec un poids de 1) au calcul. Plus précisément, ce calcul se fait de la manière suivante : pour les π_k, on compte le nombre d'observations qui ont été attribuées à la $k^{ème}$ classe; pour les θ_k^I on compte, pour chaque neurone, le nombre de séquences attribuées à la $k^{ème}$ classe et qui commencent par le neurone en question ; et pour les θ_k^T, en incrémentant les probabilités de transition d'un neurone a à un neurone b de 1 à chaque fois que, au sein d'une séquence attribuée à la $k^{ème}$ classe, les neurones a et b sont consécutifs. Enfin, après avoir effectué ceci pour toutes les séquences, on normalise les π_k, θ_k^I et θ_k^T obtenus de la même manière que ci-dessus.

5.5.1 Classification de sessions par modèle de mélange de chaînes de Markov

Pour valider l'algorithme de classification par modèle de mélange de chaînes de Markov (dont le principe a été décrit au § 5.4.3), Nous avons commencé par sélectionner 10 autoroutes parmi les 14 mentionnées au § 4.6, de manière à avoir des données plus pertinentes. En effet, l'autoroute No 9 était trop semblable à l'autoroute No 13 (ce qui aurait pu conduire à des sessions identiques

mais étiquetées par deux autoroutes différentes), l'autoroute No 10 n'était pas assez longue (seulement 2 neurones significatifs – c'est à dire différents de l'entrée et de la sortie), la 12ème autoroute était trop similaire à la 8ème, et la 14ème était également trop courte (1 seul neurone significatif). Nos autoroutes de référence étaient donc les suivants :

No Autoroute	Numéros des neurones de la séquence
1	8 17 60 40 56 41 50 57 4 54 27 31 36 42 32 62
2	8 56 36 17 41 57 50 40 9 54 4 0 62
3	8 31 41 51 56 4 5 9 0 54 57 17 1 62
4	8 17 57 41 5 9 40 4 32 62
5	8 17 56 40 57 27 41 50 9 54 32 4 1 2 62
6	8 17 56 5 4 42 41 54 32 62
7	8 17 57 54 62
8	8 27 50 42 31 5 40 9 41 56 32 4 57 0 54 2 17 1 62
9	8 17 57 9 54 1 2 4 62
10	8 9 4 62

TAB 5.1. ***TABLEAU DES AUTOROUTES REFERENCE POUR LA GENERATION DE SESSIONS ARTIFICIELLES***

A partir de ces 10 autoroutes, le générateur a créé 1000 sessions par autoroute (soit 10000 sessions en tout), avec les paramètres suivants : modifications de 35% de l'autoroute, 36% de modifications étant des suppressions de neurones, 16% des remplacements par un neurone voisin, 16% des remplacements par un neurone éloigné, 16% des insertions de neurones proches et 16% des insertions de neurones éloignés. De plus, 15% des sessions générées étaient strictement identiques à leur autoroute de référence. Enfin, un échantillon de 500 sessions a été tiré au hasard parmi les 10000 et stocké dans un fichier texte.

Les 10000 sessions ainsi générées étaient « naturellement » étiquetées (les 1000 premières par la 1er autoroute, les 1000 suivantes par le deuxième etc.), ce qui m'a permis de mesurer la qualité de la classification effectuée par l'algorithme, ainsi que de tester différents critères pour la détermination du nombre de classes (cf § 5.4.5.2 ci-dessus). Les résultats de ces essais sont décrits dans

§ 5.5.3.3 ci-dessous, et la technique – originale – d'initialisation de l'algorithme EM que nous avons proposée est présentée dans le paragraphe suivant.

5.5.2 Initialisation de l'algorithme EM

Nous avons vu que l'initialisation était une phase très importante de l'algorithme EM, puisque celui-ci converge vers un optimum *local* de la vraisemblance. Or, la méthode proposée par **[58]** pour l'initialisation nous a paru peu pertinente. En effet, celle-ci consiste à estimer les paramètres pour un modèle à une seule composante, puis à perturber ces valeurs de manière aléatoire pour obtenir K ensembles de paramètres. Or, en prenant un exemple d'application purement numérique (pour simplifier), supposons que l'on dispose de 3 classes (de cardinal égal) de nombres réels : la 1^{er} classe composée de nombres valant environ 3 , la $2^{ème}$ classe de nombres proches de 7 et la $3^{ème}$ classe de réel environ égaux à 20. Alors, la méthode de **[58]** consisterait à modéliser l'ensemble des observations par une classe de nombres proches de 10 (la moyenne de 3, 7 et 20), puis à perturber de manière aléatoire d'une petite quantité le paramètre représentatif (10) pour obtenir 3 valeurs représentatives : on pourrait alors obtenir par exemple 3 classes centrées autour de 8,9 et 13, ce qui est clairement très différent de la classification initiale.

Partant de la constatation que le taux de dissemblance calculé par DTW est une très bonne mesure de similarité entre séquences de longueur non nécessairement égales (cf résultats du § 4.6.1), nous avons eu l'idée, pour l'initialisation de l'algorithme EM, d'extraire un échantillon d'observations tirées au hasard parmi le jeu de données complet, de calculer la matrice de dissimilarité entre individus de cet échantillon en utilisant le taux dissemblance DTW (sur les séquences prises deux à deux), et d'effectuer une C.A.H. sur cette matrice. En coupant le dendogramme obtenu à un niveau tel que K classes soient mises en évidence, nous calculons ensuite les paramètres π_k, 0_k^1 et 0_k^T associés à ces K classes et utilisons ces valeurs pour initialiser l'algorithme. Précisons que, avant ce calcul, afin d'éviter que des problèmes de division par 0 dans l'équation (6) (dans §5.4.2) dans le cas où une séquence du jeu de données complet contiendrait des neurones – ou des transitions entre neurones – ne figurant dans aucune des séquences de l'échantillon (ce qui entraînerait la nullité de certains 0_k^I et/ou 0_k^T), j'ai pré initialisé les 0_k^I et les 0_k^T à une valeur constante égale à l'inverse du nombre total de neurones N_N (ce qui permet de vérifier la condition de somme des probabilités égale à l'unité). Puis, au fur et à mesure que la chaîne de

Markov associée à une composante est apprise par les séquences de sa classe associée (de l'échantillon), les valeurs des 0_k des neurones (peu ou pas du tout visités) tendent vers 0 et vice versa, les paramètres des neurones fréquemment visités croissent vers une valeur très supérieur à l'inverse du nombre total de neurones N_N.

En pratique, le temps de calcul de la matrice de dissimilarité deux à deux d'un échantillon de 500 observations était d'environ 10 minutes, ce qui était tout à fait acceptable, et j'ai donc choisi des échantillons de cette taille pour les essais. La CAH a été effectuée sur le logiciel SAS (procédures « cluster » et « tree ») en utilisant le critère de Ward pour l'agrégation des classes.

5.5.3 Résultats de test de validité de la classification

5.5.3.1 Validité des classifications obtenues par attribution douce et dure des sessions aux classes

Comme nous l'avons vu, il y a deux façons de faire fonctionner l'algorithme EM : en attribuant les sessions de manière « douce » ou « dure » aux classes (cf § 5.4.4.2). Cependant, il faut noter que, si l'attribution douce correspond à « l'esprit » original de l'algorithme tel qu'il a été décrit dans **[59]**, elle permet pas d'obtenir une partition (c'est à dire un découpage dans lequel chaque observation appartient à une et une seule classe) des individus, puisque, par construction, chaque individu est attribué à toutes les classes (avec un certain poids selon la classe). Donc, comme nous recherchions in fine à avoir une partition de la population, même dans le programme fonctionnant selon l'attribution douce, à la fin de l'algorithme (cad après la dernière itération), nous avons attribué chaque session uniquement à la classe à laquelle sa probabilité d'appartenance était maximale – alors que, selon l'attribution dure, ceci était fait à chaque itération de l'algorithme.

En pratique, les deux programmes (attribution douce et dure) fournissent en sortie deux fichiers texte : l'un contenant les classes obtenues (avec la liste des sessions de chaque classe et le taux de succès associé) et l'autre décrivant les paramètres du modèle de mélange correspondant.

Autoroutes sans cycles

En faisant fonctionner l'algorithme comme décrit ci-dessus sur 10 itérations, nous avons obtenu les taux de succès de la classification (mesurés en comptant le nombre de sessions générées à partir d'une autoroute donnée et effectivement attribuée par le programme à la classe correspondante) suivants :

No de l'autoroute / de la classe	Taux de succès (en %) Attribution douce	Taux de succès (en %) Attribution dure
1	**100 [99,62-100]**	**100[99,62-100]**
2	**100**	**100**
3	**100**	**100**
4	**100**	**100**
5	**99,9 [99,44 –99,98]**	**100**
6	**100**	**100**
7	**76,4 [73,67 –78,93]**	**92,3 [90,48-93,80]**
8	**100**	**100**
9	**100**	**100**
10	**100**	**100**

TAB 5.2. ***TAUX DE RECONNAISSANCE DES SESSIONS ARTIFICIELLES PAR LES DEUX ATTRIBUTIONS : DOUCE ET DURE.***

Ceci donne une moyenne de **97,63 % de sessions bien classées pour l'attribution douce et de 99,23% avec l'attribution dure** (les valeurs données entre crochets sont les intervalles de confiance pour un niveau de confiance de 95%, calculés d'après **[60]**.

La matrice de confusions associée, avec l'attribution dure, est la suivante :

Autoroutes	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	Total
A1	1000	0	0	0	0	0	0	0	0	0	1000
A2	0	1000	0	0	0	0	0	0	0	0	1000
A3	0	0	1000	0	0	0	0	0	0	0	1000
A4	0	0	0	1000	0	0	0	0	0	0	1000
A5	0	0	0	0	1000	0	0	0	0	0	1000
A6	0	0	0	0	0	1000	0	0	0	0	1000
A7	0	0	0	33	0	10	923	0	34	0	1000
A8	0	0	0	0	0	0	0	1000	0	0	1000
A9	0	00	0	0	0	0	0	0	1000	0	1000
A10	0	0	0	0	0	0	0	0	0	1000	1000

TAB 5.3. ***MATRICE DE CONFUSION POUR LA RECONNAISSANCE DES SESSIONS PAR L'ATTRIBUTION DURE***

D'autre part, le logarithme de la vraisemblance – on observe le logarithme plutôt que la vraisemblance elle-même car celle-ci a des valeurs trop petites pour être pratiquement interprétables-en fonction du nombre d'itérations est le suivant :

Nb d'itérations	**Log likelihood / Attribution douce**	**Log likelihood / Attribution dure**
1	-228248	-180931
2	-137323	-137472
3	-137109	-137457
4	-137076	-137457
5	-137054	-137457
6	-137039	-137457
7	-137031	-137457
8	-137027	-137457
9	-137025	-137457
10	-137023	-137457

TAB 5.4. ***PROBABILITE DE VRAISEMBLANCE PAR LES DEUX ATTRIBUTIONS : DOUCE ET DURE***

Enfin, nous avons rajouté dans le programme une procédure qui calcule les autoroutes associées aux classes obtenues de la manière suivante : pour une classe k donnée, le premier neurone de l'autoroute est celui dont le 0_k^I est maximum (comparé à celui des autres neurones), le deuxième neurone est celui dont le 0_k^I de transition depuis le premier neurone vers lui est maximum (comparé aux autres) , le troisième neurone est celui dont le 0_k^T de transition depuis le deuxième neurone vers lui est maximum etc… jusqu'à ce que le neurone obtenu soit le neurone de fin de session (ici, le neurone 62), ce qui provoque l'arrêt de la procédure. Et dans les deux cas (attribution dure et douce), les 10 autoroutes extraites de cette manière correspondaient exactement aux autoroutes de référence décrites au § 5.5.1 (TAB 5.1)

Constatant à la fois sur ces essais et sur ceux réalisés à la plus petite échelle lors de la mise au point du programme, que le taux de classification était plus élevé et la convergence plus rapide dans le cas de l'attribution dure des sessions aux classes, j'ai décidé de ne plus retenir que cette version de l'algorithme pour la suite de la validation.

Autoroutes avec cycles

Nous avons aussi testé le taux de classification obtenu en introduisant des cycles (c'est à dire l'apparition d'un même neurone deux fois) dans les autoroutes de référence, et en ré-générant 10000 sessions à partir de ces autoroutes (avec les mêmes paramètres de modification que précédemment). En l'occurrence, les nouvelles autoroutes de référence étaient alors :

No Autoroute	Numéros des neurones de la séquence
1	8 17 60 40 56 41 50 57 17 4 54 27 31 36 42 32 62
2	8 56 36 17 41 57 50 40 9 54 4 0 62
3	8 31 41 51 56 4 5 9 0 41 54 57 17 1 62
4	8 17 57 41 5 9 40 4 32 62
5	8 17 56 40 57 27 41 50 9 54 32 4 1 2 62
6	8 17 56 5 4 42 41 54 32 62
7	8 17 57 54 62
8	8 27 50 42 31 5 40 9 41 56 32 4 57 0 54 2 17 1 62
9	8 17 57 9 54 1 57 2 4 62
10	8 9 4 62

TAB 5.5. ***AUTRES AUTOROUTES DE REFERENCE***

Les autoroutes incluant des cycles étaient ainsi : la 1er (neurone 17 répété), la 3ème (neurone 41 répété) et la 9ème (répétition du neurone 57). Les taux de bonne classification obtenus ont alors été les suivants :

No de l'autoroute / de la classe	Taux de succès (en %) Attribution dure
1	100 [99,62-100]
2	100
3	100
4	100
5	100
6	100
7	93 [91,25-94,42]
8	100
9	100
10	100

TAB 5.6. ***TAUX DE RECONNAISSANCE PAR L'ATTRIBUTION DURE***

Finalement, dans les deux cas – autoroutes avec et sans cycles – les probabilités marginales d'appartenance aux classes (c'est à dire les $\pi_{k)}$ étaient toutes environ (à quelques % près) égales à 0,1.

5.5.3.2 Indépendance de la classification vis à vis de l'échantillon initial

J'ai ensuite cherché à vérifier que la classification du jeu de données entier effectuée par le programme était indépendante de l'échantillon (de 500 observations) utilisé pour l'initialisation de l'algorithme.

Pour ceci, j'ai comparé les classifications obtenues sur le même ensemble de départ (de 10000 sessions), mais en prenant quatre autres tirages aléatoires différents pour l'échantillon sur lequel la CAH était effectuée. Les taux de succès de classification ont alors été les suivants :

Autoroute / classe	**Succès (%) Echant No 2**	**Succès (%) Echant No 3**	**Succès (%) Echant No 4**	**Succès (%) Echant No 5**
1	100	100	100	100
2	100	100	100	100
3	100	100	100	100
4	100	100	100	100
5	100	100	100	100
6	100	100	100	100
7	95,4	90,3	91,6	85,9
8	100	100	100	100
9	100	100	100	100
10	100	100	100	100

TAB 5.7. ***TAUX DE RECONNAISSANCE POUR PLUSIEURS TYPES D'ECHANTILLONS D'INITIALISATION***

Ceci correspond à un taux moyen (sur toutes les classes) de bonne classification de 99,54%, 99,03%, 99,16% et 98,59% (respectivement pour les échantillons No 2,3,4 et 5).

D'autre part, j'ai également comparé les valeurs des paramètres des dix chaînes de Markov obtenues pour ces quatre échantillons à celles du premier échantillon. Sans rentrer dans le détail des comparaisons de tous les paramètres pour tous les échantillons (ce qui serait fastidieux et sans grand intérêt), j'en résumerai le résultat comme suit :

- environ 5 chaînes de Markov sur les 10 ont exactement les mêmes paramètres ($\pi_{k,}0_k^1$ et 0_k^T) d'une classification à une autre
- pour les 5 chaînes restantes d'un modèle à un autre les π_k et 0_k^I et 0_k^I sont quasiment (à 1 ou 2 % près) égaux, et les 0_k^T sont très similaires, cad que les $(0_k^T)_{ij}$ sont strictement identiques pour entre 45 et 50 valeurs de i (sur les 53 neurones actifs) et que pour les autres valeurs, ils sont semblables (cad que dans 95% des cas, ils sont égaux à 50% près).

5.5.3.3 Résultat de test de critères pour la détermination du nombre de classes

- Compte tenu du grand nombre de critères de détermination du nombre de classes optimal disponibles, la question était de trouver lequel était le plus adapté à notre problème (la classification par modèle de mélange (la classification par modèle de mélange de chaînes de Markov).

- Les deux critères les plus utilisés étant l'AIC et le BIC, on s'est restreint à leur étude, ainsi qu'à celle du critère proposé par **[58]** (qu'on notera ici le « CIC », pour « Cadez Information Criterion » !).

- J'ai donc calculé les valeurs de ces trois critères pour différentes valeurs de K (à la fois dans la CAH de l'échantillon initial et dans le modèle calculé par l'algorithme), pour les mêmes données que précédemment (10000 sessions générées à partir des 10 autoroutes sans cycle, échantillon No 1), et avec la valeur suivante pour le nombre de paramètres indépendants à estimer $\nu(M)$ du modèle M :

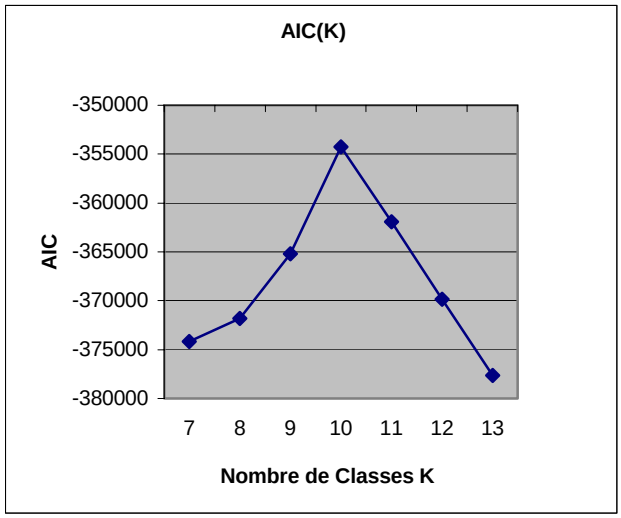
- $\nu(M)$ = (Nombre de π_k indépendants) + Kx(Nombre de 0_k^1 indépendants) + Kx(Nombre de 0_k^T indépendants)

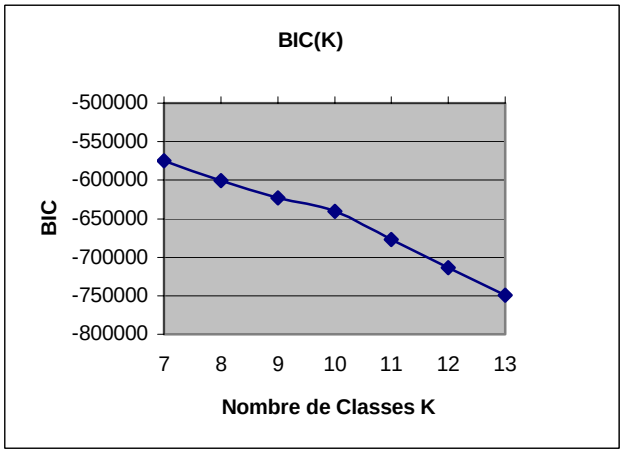
  - $=(K-1) + Kx(N_N-1) + Kx(N_N-1) \times N_N$

, où N_N est le nombre total de neurones, et en vertu des conditions sur les probabilités :

$\sum_k \pi_k = 1$, $\sum_{\text{neurones}} 0_k^I = 1$ et $\sum_{\text{neurones j}} (0_k^T)_{ij} = 1$ (à neurone i fixé).

Les graphiques des AIC(K), BIC(K) et CIC(K) obtenus ont alors été les suivants :





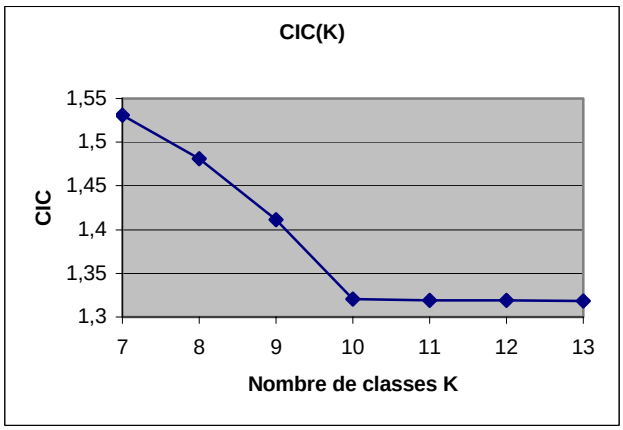


FIG 5.1. ***DIFFERENTS CRITERES D'INFORMATIONS POUR LE CALCUL DU NOMBRE DE CLASSES***

5.5.4 Discussion, analyse et interprétation des résultats

5.5.4.1 Reconnaissance

Il est clair qu'une validation de la reconnaissance se basant uniquement sur le critère d'achat est par construction grossière puisque sur les 14 autoroutes, il y en a 6 « Achat » et 8 « Non-Achat ». Une session « Achat » peut donc par exemple être attribuée à une autoroute « Achat », et être ainsi considérée comme bien reconnue alors qu'en fait une autre autoroute « Achat » en serait plus similaire.

En gardant à l'esprit cette réserve, on peut néanmoins observer les résultats obtenus une supériorité de l'algorithme de DTW sur le calcul de distorsion. En effet, si les taux de reconnaissance correcte du comportement « NAC » est le même pour les deux méthodes (99%), le taux d'erreur dans la reconnaissance du comportement « AC » est environ 15% plus élevé (en valeur relative) avec le calcul de distorsion qu'avec le DTW (et 4% en valeur relative).

5.5.4.1.1 Comparaison DTW / QV pour les données artificielles

Dans le cas de données générées artificiellement, le problème précédent ne se pose plus et la validité de la reconnaissance peut être mesuré de manière beaucoup plus précise, puisqu'on sait d'avance exactement quelle autoroute « étiquette » chaque session .

Les résultats obtenus montrent alors une supériorité très nette de l'algorithme de DTW sur le calcul de distorsion :

- Le taux de reconnaissance est entre 18 et 53% plus élevé en valeur relative et entre 15 et 34% en valeur absolue, selon le type de sessions générées.
- Il est, en moyenne sur toutes les sessions générées, 36% plus élevé en valeur relative et 26% en valeur absolue
- Quelque soit l'autoroute, le taux de bonne reconnaissance est systématiquement supérieur (voire très supérieur pour certains), et est toujours au moins de 70% pour la DTW alors qu'il peut tomber jusqu'à 0% (autoroute No 4) avec la calcul de distorsion.

Nous attribuons cette supériorité de la reconnaissance par DTW à celle par calcul de distorsion à deux facteurs :

- D'une part au codage par poids de position, qui induit une perte d'information sur l'ordre entre les neurones dans les séquences

- D'autre part au fait que le DTW tient compte de la proximité entres les neurones sur la carte – puisqu'il travaille sur les vecteurs à 82 composantes qui représentent les neurones - , ce que ne fait pas le calcul de distorsion

5.5.4.2 Classification

5.5.4.2.1 Validité de la classification

Analyse

On constate que la classification obtenue est très bonne dans le cas de l'attribution douce des séquences aux classes (97,63% de sessions bien classées en moyenne) et encore meilleure (99,23% de réussite) avec l'attribution dure. La seule autoroute pour laquelle le taux de classification n'a pas été de 100% (ou 99,9%…) est la No 7.

D'autre part, dans le premier cas, l'algorithme converge relativement lentement et, même après 10 itérations, le logarithme de la vraisemblance continue d'augmenter ; alors que dans le deuxième cas, le programme converge très rapidement (au bout de 3 itérations) vers une valeur constante de la « Log Likelihood ».

Discussion et interprétation

Nous attribuons la qualité de ces résultats d'une part à notre technique d'initialisation de l'algorithme EM (cf § 5.5.2), qui fait que celui-ci démarre à partir de « bonnes » valeurs des paramètres 0, et d'autre part au nombre relativement élevé (plus de 50) d'états possibles des chaînes de Markov (c'est à dire de neurones), qui autorise une grande finesse de distinction entre les séquences.

Le classement de quelques sessions issues de l'autoroute No 7 dans les classes associées aux autoroutes No 4 et No 9 n'est pas surprenante : nous l'attribuons d'une part à la petite longueur de l'autoroute No 7 – seulement 3 neurones significatifs – et d'autre part au fait que ses premiers neurones (8,17,57) sont identiques à ceux des autoroutes No 4 et 9 :

En effet, le fait que l'autoroute – et donc les sessions générées grâce à elle – soient courtes donne moins d'informations à l'algorithme pour effectuer la classification. L'autoroute No 10 est également courte – 2 neurones significatifs -, mais si courte que les sessions générées à partir d'elle lui sont strictement identiques, car le nombre de neurones de l'autoroute modifiés par le générateur est la partie entière de $2 \times 35 / 100$, soit 0 (alors que pour l'autoroute No 7, le

nombre de neurones modifiés est la partie entière de 3 × 35 / 100, soit 1).

De plus, la similarité des neurones de l'autoroute No 7 avec ceux du début des autoroutes No 4 et 9 implique que les 0_k^I et les 0_k^T des chaînes de Markov qui leur sont associées soient également proches, d'où les erreurs constatées dans la matrice de confusion.

D'autre part, nous ne sommes pas non plus surpris de la qualité équivalente des classifications obtenues générées à partir d'autoroutes avec et sans cycles. En effet, généralement, pour une autoroute sans cycle, une fois l'apprentissage des paramètres terminé, la probabilité de transition d'un neurone de l'autoroute à une autre est de l'ordre de 0,8, tandis que les probabilités de transition entre deux neurones non consécutifs de l'autoroute sont très inférieurs à 0,1 (et encore plus faibles dans le cas de neurones n'appartenant pas à l'autoroute). L'adjonction d'un cycle se divise approximativement par deux la probabilité de transition à partir du neurone répété, qui devient donc alors environ égale à 0,4. Donc, toutes choses étant égales par ailleurs, l'ajout d'un cycle fait que, dans le calcul des probabilités d'appartenance des sessions aux classes (d'après l'équation (6)), le 0,8 ci-dessus est remplacé par environ $(0,4)^2$, i.e. 0,16, soit un nombre toujours supérieur aux probabilités de transition entre neurones non consécutifs de l'autoroute ou n'en faisant pas partie. La probabilité d'appartenance à la classe correspondante à l'autoroute est donc presque toujours supérieure à celle des autres autoroutes, et la classification finale est donc quasiment la même.

Enfin, le fait que les π_k trouvés soient tous environ à 0,1 est cohérent avec la partition de nos données de validation en dix classes de taille égale.

Nous n'avons en revanche pas trouvé d'explication théorique au fait que la classification par attribution dure des sessions aux classes soit meilleure (et, qui plus est, obtenue plus rapidement !) que par l'attribution douce. Nous nous contentons donc de le constater empiriquement … et de l'exploiter !

Nous n'avons pas pu comparer ces classifications à celle qui auraient été obtenues par les techniques décrites au § 4, à cause d'une panne de l'ordinateur qui contenait les programmes correspondants. Néanmoins compte tenu d'une part de la perte d'information induite par le codage par « poids de position », que nous avons constatée dans le § 4.6.3, et d'autre part du taux de succès extrêmement élevé

(plus de 99%) de notre méthode, nous sommes persuadés que cette dernière se serait avérée supérieure.

5.5.4.3 Indépendance de la classification vis à vis de l'échantillon initial

On constate que les classifications obtenues en prenant cinq échantillons différents pour l'initialisation de l'algorithme EM sont quasiment les mêmes. Ceci est très satisfaisant et est une validation supplémentaire la qualité de notre technique d'initialisation.

Nous attribuons les quelques différences significatives entre certaines probabilités de transition d'une classification à une autre au fait qu'elles correspondent à des transitions entre des neurones qui, lors de la génération aléatoire, apparaissent rarement dans les sessions de départ. Ces transitions peuvent alors apparaître par exemple 4 fois dans échantillon et 2 fois dans une autre, engendrant une différence de 100% dans les probabilités de transition des chaînes de Markov associées à l'initialisation. Les transitions apparaissant rarement dans le jeu complet de données, le déroulement de l'algorithme ne suffit alors pas à faire converger les θ_k^T associés aux deux échantillons (pour ces transitions) vers des valeurs proches.

5.5.4.4 Choix du critère pour la détermination du nombre de classe optimal

Analyse

Sur les graphiques des AIC(K), BIC(K), CIC(K), on voit que :

- AIC croît régulièrement avec K quand celui-ci varie entre 7 et 10, puis décroît de manière à peu près symétrique quand K augmente de 10 à 13. AIC atteint donc un maximum très nettement observable en K= 10.
- BIC décroît régulièrement avec K quand celui-ci varie entre 7 et 13, et n'atteint donc de maximum pour aucune valeur de K dans cet intervalle (sauf évidemment en K=7).
- CIC décroît régulièrement pour les valeurs de K comprises entre 7 et 10, puis décroît beaucoup plus lentement quand K passe de 7 à 13. On observe donc clairement un point d'inflexion en K=10.

Discussion et interprétation

Étant donné que les données étaient partitionnées en 10 classes, la qualité essentielle que nous cherchions à mettre en évidence pour ces critères était le fait qu'ils présentent un optimum en K=10.

Le AIC est le seul des trois critères ci-dessus à avoir vérifié cette propriété, et il apparaît donc clairement comme le plus adapté à ce type de classification.

A contrario, le BIC, en décroissance constante, serait complètement inadapté.

Le CIC semble être dans une situation intermédiaire. En effet, bien qu'il ne présente pas d'optimum (ce que, en toute rigueur, nous recherchions), sa pente change de manière si significative à partir de K=10 qu'on pourrait envisager de l'utiliser, par exemple en cherchant par calcul numérique la valeur de K pour laquelle la dérivée seconde de CIC devient proche de 0 ou, plus simplement, pour laquelle le pourcentage de décroissance de CIC chute brutalement.

Néanmoins, ceci est plus compliqué et comporte plus de risques d'erreur que la simple observation d'un maximum permise par le AIC, aussi nous concluons que ce dernier est le meilleur pour notre problème.

5.6 Conclusion

Nous avons étudié dans ce chapitre, la modélisation par mélange de chaînes de Markov. Cette modélisation a été appliquée à la classification des sessions de navigation, en prenant comme états des chaînes de Markov les neurones de la carte topologique précédente. Une nouvelle méthode d'initialisation de l'algorithme d'apprentissage des paramètres du modèle (l'algorithme EM) a été introduite. Le principe en est de calculer les paramètres initiaux à partir des classes obtenues par la classification ascendante hiérarchique (CAH) d'un échantillon de sessions dont nous avons calculé la matrice de dissimilarité par la programmation dynamique (DTW).

Chapitre VI

6 Classification Évolutive : Restructuration

Les techniques de classification considèrent que la base de données traitée est complètement représentative pour le problème, alors que c'est rarement le cas avec les problèmes complexes. Par conséquent, il serait très utile d'enrichir la base pour prendre en compte les formes qui n'auraient pas été disponibles lors de la constitution de la base. Le problème serait donc de mettre à jour la classification en introduisant un processus d'évolution.

6.1 Introduction

Les techniques numériques de la reconnaissance des formes reposent sur des méthodes de classification robustes, pour comprendre et simplifier les gigantesques bases de données multidimensionnelles. Dans la plupart du temps, K-means et d'autres méthodes de partitionement sont utilisés pour des applications industrielles. Cependant, leurs solutions finales dépendent sur leur initialisation, particulièrement sur le nombre de classes. Or, cette connaissance préalable est rarement disponible pour l'utilisateur.

6.2 État de l'art

Nous allons aborder dans cette section, plusieurs méthodes reflétant les efforts déjà élaborés au sujet de la classification évolutive. L'objectif de cet état de l'art, est de s'inspirer des points forts de ces méthodes, pour les utiliser dans la méthode que nous proposons, et qui sera traitée dans les sections qui suivent.

6.2.1 Classification hiérarchique évolutive

Cette méthode **[49]** a été introduite à cause des limites de la classification hiérarchique classique. Tout en sachant que cette dernière, fournit une bonne représentation des données avec différentes possibilités de partitionement, elle souffre considérablement du problème de la mémoire avec une augmentation exponentielle par rapport au nombre de données de la base. Par exemple pour traiter 10000 éléments, il faut compter 200 Mo de RAM. D'où son impuissance pour les importantes applications industrielles.

Pour cette raison, une stratégie a été élaborée et consiste à construire une première taxonomie utilisant seulement la mémoire disponible et la mettre à jour à l'arrivée de nouveaux éléments. Le succès de cette stratégie dépend de la capacité de mettre à jour la taxonomie tant qu'on garde sa plus large part possible.

6.2.1.1 Détermination de la place du nouvel élément dans la hiérarchie

Pour ajouter un nouvel élément dans une taxonomie, on doit déterminer sa position dans l'arbre. Plusieurs méthodes ont été proposées à ce sujet . Elles sont généralement basées sur l'estimation d'une couche frontière entre les deux principaux sous-arbres (i.e. groupes). Chaque couche frontière est utilisée pour positionner le nouvel élément (NE) comme l'aurait fait un arbre de décision. Cependant, à notre connaissance, aucune de ces méthodes n'est contrainte par un critère d'arrêt .

Il est bien sûr impossible de considérer que le NE pourra être le frère de son plus proche voisin. En plus, la distance minimale moyenne entre NE et un groupe peut être atteinte pour un groupe qui ne contient pas les plus proches voisins de NE. Donc la recherche doit commencer à partir de la racine de la taxonomie. Cela implique la définition d'une procédure descendante dans l'arbre (incluant une condition d'arrêt) **[49]**.

Le principe consiste à utiliser la notion de région d'influence (ROI) d'un groupe G. Soit $D(G_1,G_2)$ la distance entre les deux groupe G_1 G_2. Donc, comme le montre FIG 4.1, un point x devient le ROI de G_1 si $D(x, G_1) \leq D(G_1, G_2)$.

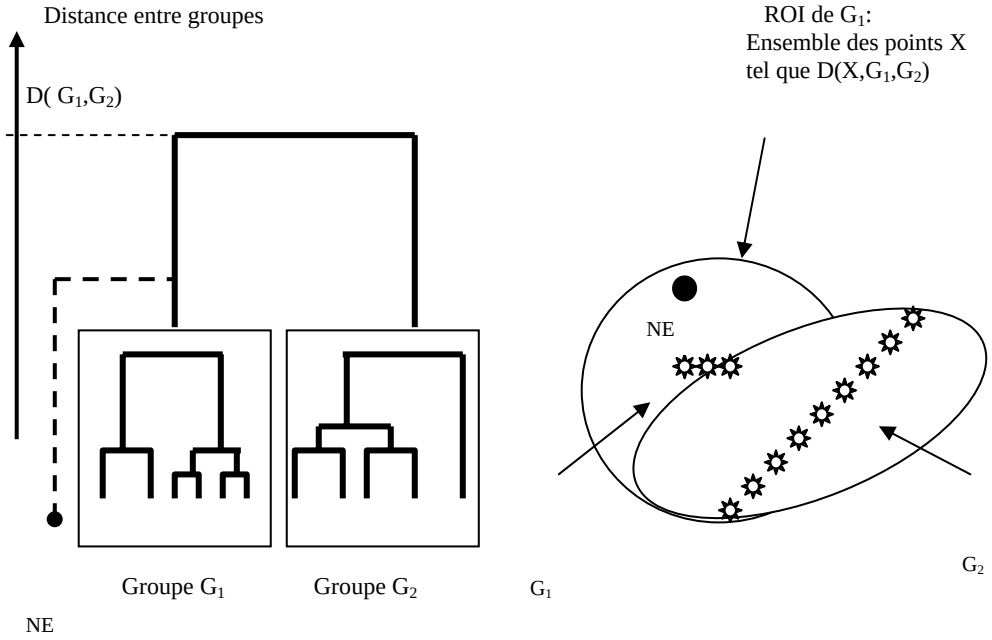


FIG 6.1. ***RECHERCHE DE LA PLACE DU NOUVEL ELEMENT DANS LA TAXONOMIE***

6.2.1.2 La mise à jour de la taxonomie

La seconde phase de l'algorithme consiste à mettre à jour les nœuds de la taxonomie, commençant par le point précédemment déterminé. Le principe consiste à couper le sous-arbre seulement en cas de nécessité, et de mettre à jour les niveaux des nœuds dans les autres cas. Le problème est de déterminer les conditions à vérifier pour couper un sous – arbre.

Deux possibilités sont envisageables pour mettre à jour la taxonomie **[49]**. Dans la première, on pourra essayer de déterminer les changements exacts sur la taxonomie dû à la présence du nouvel

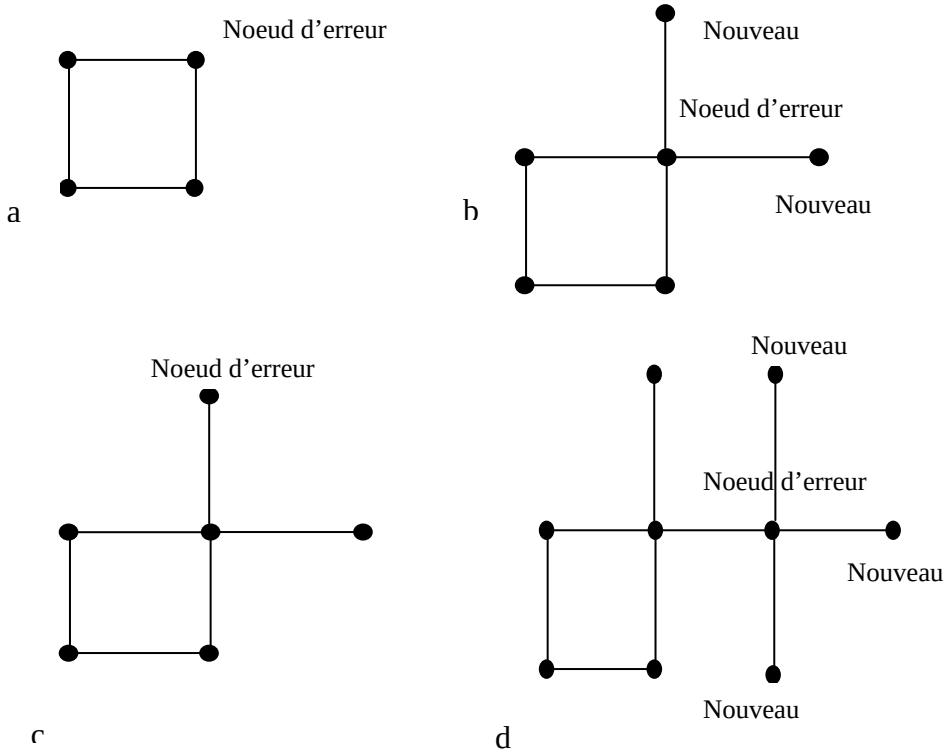
élément. Dans le deuxième cas, on pourra déterminer la partie de la taxonomie qui ne pourra jamais changer après l'addition du NE.

6.2.2 La méthode IGG (Incremental Grid Growing)

L'algorithme IGG **[50]** est une amélioration de l'algorithme SOM. Il s'agit d'une nouvelle technique qui permet à la structure du réseau d'être dynamique et adaptative.

Initialement, la grille est constituée de quatre (4) nœuds de connexions avec les vecteurs de poids (les prototypes). Ces vecteurs sont choisis d'une manière aléatoire dans l'espace d'entrée. Les trois étapes suivantes sont faites de manière itérative :

- Adapter la grille courante à travers le processus SOM.
- Ajouter les nouveaux nœuds au périmètre de la grille, quand le réseau montre une grande erreur dans la représentation.
- Examiner tous les vecteurs de poids des nœuds voisins, pour déterminer si une connexion entre deux nœuds est supprimée ou une nouvelle connexion est ajoutée. La FIG 6.2 montre le processus de l'algorithme IGG :



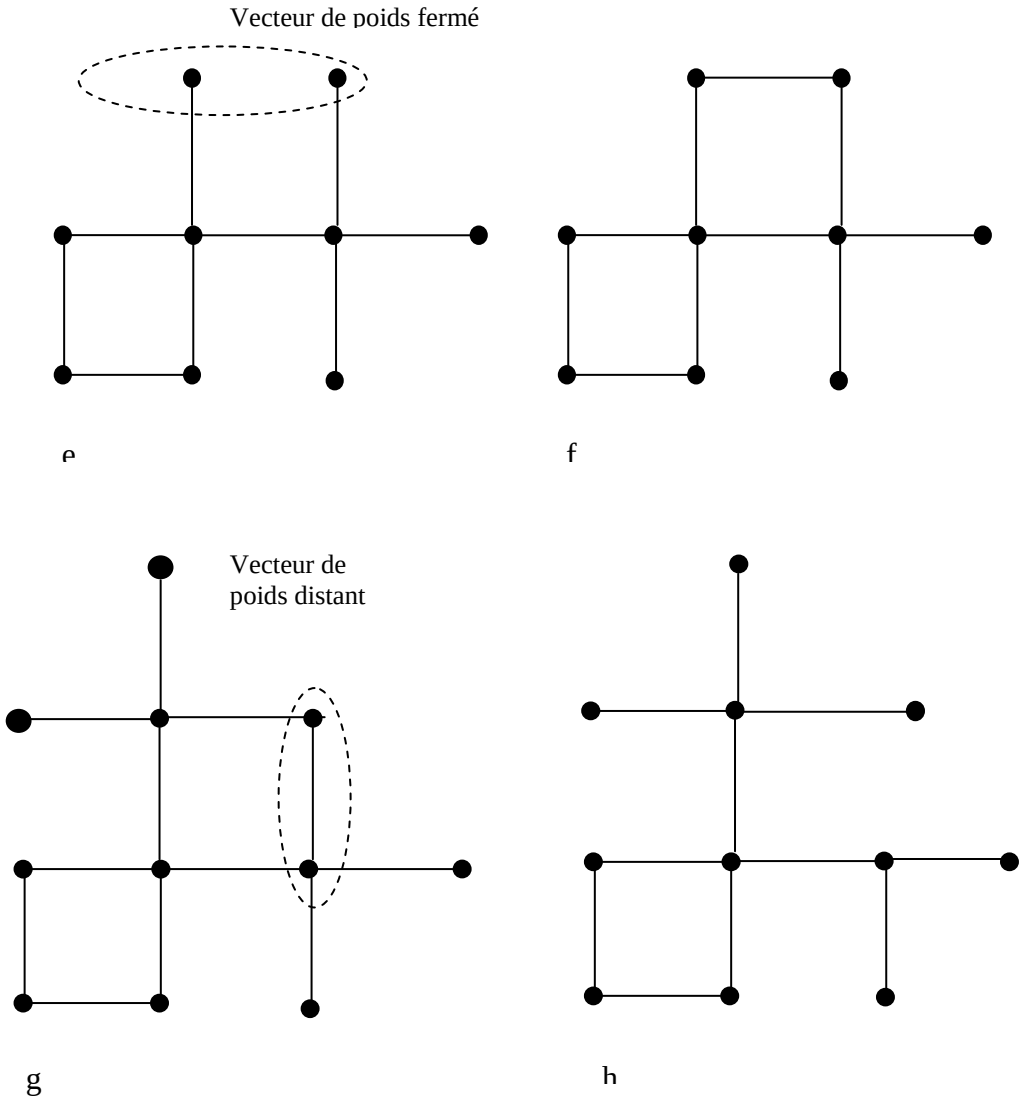


FIG 6.2. ***LE PROCESSUS IGG***

- FIG 6.2a : montre la structure initiale après la première organisation. Le nœud à l'extrémité avec la plus grande valeur d'erreur est marqué.
- FIG 6.2b : les nouveaux nœuds sont développés dans la grille dans le voisinage du nœud d'erreur.
- FIG 6.2c : après l'organisation de la nouvelle structure par le processus SOM, un nouveau nœud d'erreur est trouvé.
- FIG 6.2d : de nouveaux neurones sont ajoutés dans la grille, dans le voisinage immédiat du nœud d'erreur.

- FIG 6.2e : pendant l'organisation autonome de cette nouvelle structure, l'algorithme détecte que le nœud développe un vecteur de poids très fermé en fonction de la distance euclidienne.
- FIG 6.2f : le nœud « fermé » est connecté.
- FIG 6.2g : après une autre organisation, l'algorithme découvre une connexion entre 2 nœuds qui n'est pas convenable avec la représentation de l'entrée.
- FIG 6.2h : le nœud « distant » est déconnecté de la grille.

Nous pouvons remarquer que, bien que le processus soit évolutif, il utilise toujours la même base à chaque adaptation de la grille courante. L'aspect évolutif ici, concerne donc, la topologie sans s'occuper de l'arrivée dynamique des données.

6.2.3 La méthode GNG (Growing Neural Gas)

Cette méthode **[52]** est une variante à sa précédente (Neural Gas **[51]**) qui propose d'adapter les prototypes de SOM indépendamment d'un arrangement topologique prédéfini sur l'hypothèse que l'information sur l'arrangement des prototypes est implicitement donnée par les distorsions associées à chaque vecteur d'apprentissage.

La version évolutive consiste à la fois à créer les unités (neurones) et les connexions topologiques.

Le processus de cette version procède de la manière suivante :

- 1. Initialisation de deux neurones c_1, c_2 dans l'espace : $w_{ci} \in R^n$

$$A=\{ c_1,c_2\}$$

$$C \subset A \times A,\ C = \phi$$

- 2. Présentation d'un exemple d'apprentissage : x
- 3. Détermination du 1er gagnant s_0, et le 2ème gagnant s_1, par rapport à x

$$s_0 = \underset{c \in A}{\operatorname{argmin}} \| x - w_c \|$$

$$s_1 = \underset{c \in A / \{s_0\}}{\operatorname{argmin}} \| x - w_c \|$$

- 4. La création de connexion entre s_0 et s_1 en cas de son absence

$$C = C \cup \{(s_0, s_1)\}$$

$$ts_0 s_1 \leftarrow 0$$

- 5. Incrémentation de l'erreur associée au 1er gagnant s_0

$$\Delta Es_0 = \|x - w_{s0}\|^2$$

- 6. Adaptation des prototypes

$$Ws_0(t+1) = Ws_0(t) + \alpha_b(x - Ws_0(t)$$

$$Wi(t+1) = Wi(t) + \alpha_n(x - Wi(t), \ \forall i \in Ns_0$$

- 7. Incrémentation de l'âge des connexions partant du 1er gagnant

$$ts_{0i} \leftarrow ts_{0i} + 1, \ \forall i \in Ns_0$$

- 8. Suppression des connexions dont l'age dépasse un seuil temporel fixé

$$ts_0 j > T$$

supprimer les neurones ne possédant pas de connexions

- 9. Détermination du neurone (q) possédant la plus grande erreur accumulée

$$q = \underset{c \in A}{\operatorname{argmax}} E_c$$

- 10. Détermination, parmi les voisins du neurone trouvé précédemment, du neurone (g) ayant la plus grande erreur

$$g = \underset{c \in N_q}{\operatorname{argmax}} E_c$$

- 11. Ajout d'un neurone (r) au réseau et initialisation de son prototype par rapport aux deux neurones trouvés précédemment

$$A = A \cup \{r\}, \ w_r = (w_q + w_g)/2$$

- 12. Connexion de r à q et suppression de la connexion entre q et g

$$C = C \cup \{(r,q),(r,g)\}, \ C = C \setminus \{(q,g)\}$$

- 13. Diminution de l'erreur de q et de g

$$\Delta E_q = -\delta E_q, \ \Delta E_g = -\delta E_g$$

- 14. Approximation de l'erreur locale pour r

$$E_r = (E_q + E_g)/2$$

- 15. Diminution des erreurs locales pour tous les neurones

$$\Delta E_c = -\gamma E_c, \ \forall c \in A$$

- 16. Arrêt du processus (taille du réseau, mesure de performance,…) sinon Retour à la deuxième étape.

La figure ci – dessous montre quelques étapes d'une simulation d'une distribution de données sur une figure circulaire.

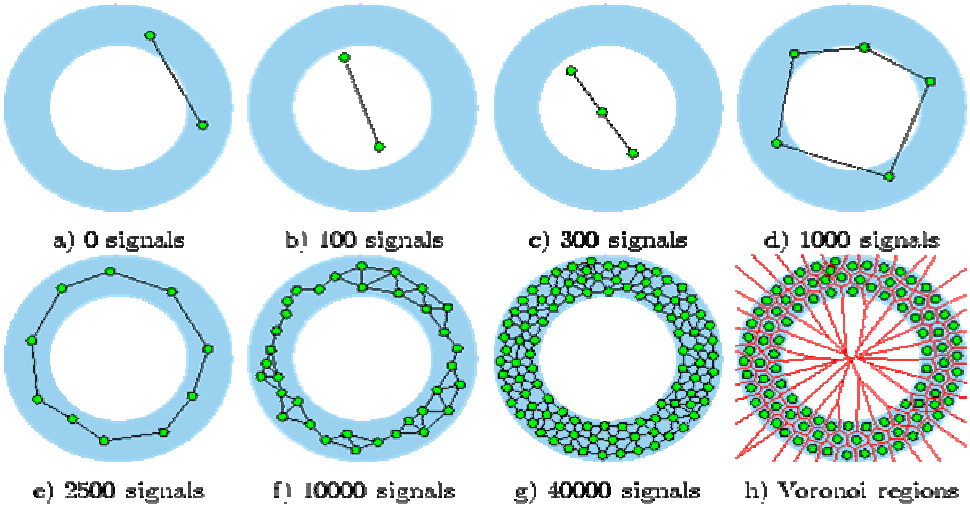


FIG 6.3. UNE SEQUENCE DE SIMULATIONS DE L'ALGORITHME GNG SUR UNE DISTRIBUTION DE PROBABILITES UNIFORME SUR UNE FIGURE CIRCULAIRE

Cette méthode souffre d'un problème de lissage. Cela veut dire que, après sa création, la carte est représentée par des sous-ensembles de neurones indépendants (FIG 6.4). En effet, entre chaque paire de sous-ensembles, des neurones ne possédant pas de connexions (étape 8 dans l'algorithme ci-dessus), sont définitivement supprimés. Cela dit, ces neurones n'auront jamais la possibilité d'être activés par un éventuel ensemble de données qui risquent d'arriver ultérieurement.

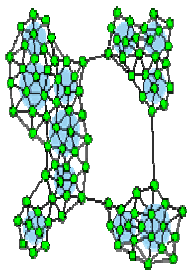


FIG 6.4. ***UN EXEMPLE DE REPRESENTATION DES NEURONES DE LA CARTE ISSUE DE L'ALGORITHME GNG.***

6.3 Un bref rappel sur les cartes auto-organisatrice (SOM)

L'apprentissage dans les cartes topologiques se fait avec une fonction de voisinage. Il procède en 3 étapes :

- *Initialisation* : il s'agit de l'initialisation des poids.
- *Compétition* : à chaque entrée d'un exemple au réseau, un calcul de distance est effectué pour activer un neurone dit gagnant , c'est celui dont le potentiel d'activation est le plus fort en fonction de l'entrée.
- *Adaptation* : le choix d'un nœud particulier permet alors d'ajuster les poids localement, en minimisant la différence qui existe encore entre les poids et le vecteur d'entrée. Cet ajustement se fait suivant une forme de voisinage qui peut être carrée, ronde ou hexagonale.

Le processus est donc constitué de ces trois phases qui sont itérées jusqu'à la minimisation d'une erreur globale calculée sur l'ensemble du réseau ou sur un nombre de cycles d'apprentissage fixé empiriquement.

6.4 Une version évolutive pour les cartes topologiques

Dans cette section, nous proposons une nouvelle version des cartes topologiques. Cette version possède un aspect évolutif « complet ». En effet, il s'agit de créer la topologie en fonction de l'arrivée des données. Nous appelons cette version e-SOM.

Notre problématique est due à l'arrivée de plusieurs masses de données après la création du modèle initial. Au lieu de refaire tout le modèle, nous proposons de garder l'existant est de l'incrémenter (le mettre à jour), en fonction des nouvelles vagues de données qui arrivent.

6.4.1 La gestion des nouveaux neurones

Nous proposons tout d'abord d'étudier la distribution des nouvelles données sur la carte initiale. Pour cela nous appliquons la procédure de *compétition* de SOM (i.e. le classement des données dans la carte). Nous trouvons, par exemple, 10% de données sur le premier neurone, 24% sur le $i^{éme}$ neurones…etc

De manière statistique, nous marquons les neurones qui appartiennent au périmètre de la grille et qui possèdent une densité (effectif) supérieur ou égal à $((C/N)*100)\%$. Tels que C représente le nombre des neurones de la carte et N le nombre d'exemples dans la nouvelle base en cours.

Intuitivement, nous ne nous intéressons qu'aux neurones du périmètre, car nous estimons que les neurones situés à l'intérieur de la carte, sont déjà enfermés par leur voisinage, alors que ceux des frontières possèdent une possibilité de voisinage en dehors de la carte. Par exemple, le neurone en bas à gauche, possède deux possibilités de voisinage (un au dessous et un autre à gauche)

De plus, l'ajout des neurones dépend du voisinage du neurone marqué (un neurone est dit marqué, s'il est actif et autorisé à un voisinage).

Remarque : *un neurone marqué n'est pas forcément réel (nous verrons l'explication dans la section 6.4.3)*

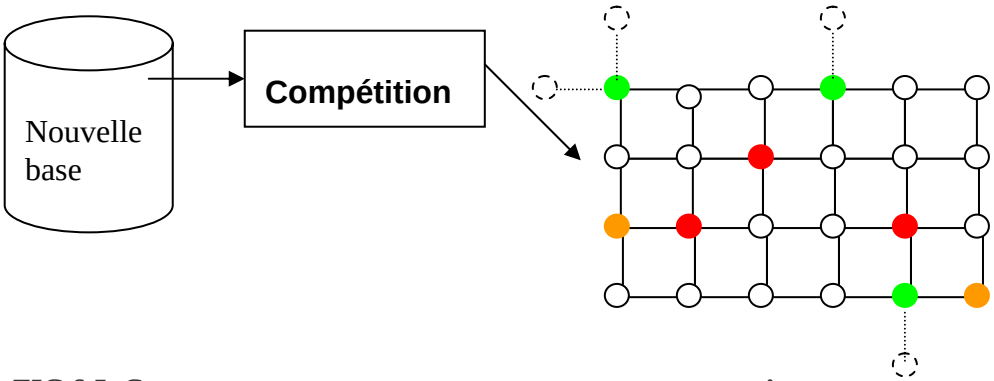


FIG 6.5. ***GESTION DES NOUVEAUX NEURONES AJOUTES POUR L'APPRENTISSAGE EVOLUTIF :***

- *Neurone actif autorisé à avoir de nouveaux voisins*
- *Neurone actif non marqué , car sa densité n'est pas importante*
- *Neurone actif mais non autorisé à un nouveau voisinage à cause de sa position dans la carte*
- *Nouveau neurone « provisoirement » créé.*

La deuxième question qui se pose est la suivante:

Comment initialiser les nouveaux neurones créés ?

Nous proposons donc l'initialisation suivante :

$$W^*_j(k) = \frac{1}{2}\left(\frac{1}{N'}\sum_{i=1}^{N'} V_i(k) + W_j(k)\right) + \varepsilon$$

N_c : est le nombre de neurones de la carte initiale

$W^*_j(k)$: est le $k^{ème}$ élément du vecteur poids du nouveau neurone créé au voisinage du neurone *j(j est le neurone marqué)*

N' : est l'effectif de la nouvelle base

Vi(k) : est le $k^{ème}$ élément du $i^{ème}$ vecteur de la nouvelle base.

$W_j(k)$: est le $k^{ème}$ élément du vecteur poids du neurone *j* après l'apprentissage de la première carte.

ε : est un nombre aléatoire tel que : $0 \leq \varepsilon \leq 1$.

6.4.2 La mise à jour de la topologie

Une fois le nouveau voisinage organisé, il ne reste qu'à entraîner la carte avec la nouvelle base. Nous répétons donc, le processus SOM sur la nouvelle base pour l'adaptation générale sur les anciens et les nouveaux neurones. Cela fait référence à un apprentissage par morceau (de neurones de la carte) en fonction des données disponibles.

Le processus évolutif est itératif à chaque arrivée de nouvelles bases de données, ce qui explique la dynamique de la restructuration de la carte.

6.4.3 L'élimination des nouveaux neurones

Une fois l'apprentissage évolutif fini à chaque arrivée, on réutilise la procédure de compétition de SOM pour déterminer l'activation des neurones marqués selon le principe de §6.4.1.

Si après cette compétition, le neurone créé est inactif , il est directement supprimé sinon il est gardé , et il devient donc « réel ».

Nous présentons la formulation algorithmique qui interprète cette nouvelle version :

1. Application de SOM (Création de la première carte: C_0) sur la première base de données (BDD_0)
   a. Initialisation des poids
   b. Compétition des formes par rapport au poids
   c. Adaptation des poids

2. Mise à jour de la topologie (i=1)
   a. $\forall z \in BDD_i$, $\forall j \in C_0$, Compétition (z, j).
   b. Si $j \in$ périmètre (C_0) Alors
      Si proportion (j) $\geq$ (100×C/N) % Alors
      Création (j , voisinage(j))
      Finsi
      Finsi

3. Initialisation des nouveaux neurones

$$W^*_{j'}(k) = \frac{1}{2}\left(\frac{1}{N'} \sum_{i=1}^{N'} V_i(k) + W_j(k) \right) + \varepsilon$$

4. Adaptation
   a. Adaptation selon SOM de C_i , $\forall j$
   // $C_I = C_{I-1} \cup \{J\}$//

5. Remise à jour
   a. Éliminer tout j' , tel que Compétition (j', C_i) = $\varnothing$

6. i = i+1, si i < M Alors Retour à 2 sinon Arrêt
   // M : représente le nombre de sous – bases.

6.5 Application : mise à jour du modèle de visualisation

Il faut mentionner que nous avons développé une version évolutive de SOM sur les données Internet car nous étions confronté au problème d'arrivée continue des fichiers Log. Comme la première cartographie du site était significative, nous avons proposé de la garder et de la mettre à jour en fonction des nouvelles interactions dans le site.

Nous avons appliqué notre version évolutive sur l'ensemble des bases. Par conséquent, notre première constatation était faite sur le nombre de neurones de la carte (FIG 6.5).

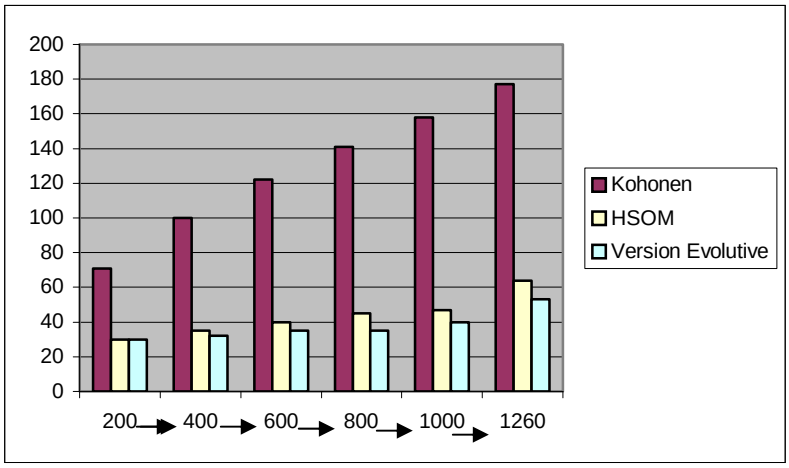


FIG 6.5. ***COMPARAISON ENTRE LES TROIS METHODE SUR LE NOMBRE DE NEURONES DE LA CARTE***

Dans la figure ci – dessus, la version évolutive s'avère la plus optimale en nombre de neurones créés selon le nombre d'exemples d'apprentissage. De plus, cette version bénéficie des avantages suivants :

- Création de la carte à partir d'une topologie initiale créée à partir d'une base réduite
- Préservation de la notion du lissage qui consiste à garder la relation entre les neurones actifs et les neurones inactifs

Les flèches dans la figure montrent l'incrémentation successive de la base en ajoutant 200 exemples après chaque apprentissage.

A cause de la non disponibilité des fichiers Log du même site traité auparavant, nous avons pensé à diviser la première base (issue du premier fichier Log) en plusieurs bases soit 6 bases. La figure FIG 6.6 montre la cartographie du site issue de la version évolutive que nous avons apportée à l'algorithme SOM. Cette figure est optimale par rapport à sa précédente. De plus, l'algorithme permettant de l'obtenir, présente de meilleurs résultats sur la décroissance de l'erreur quadratique ainsi que l'erreur topologique (TAB 6.1).

Méthode	Taux de décroissance	Erreur topologique
SOM	0.30	0.3
e-SOM	.36	0.1

TAB 6.1. COMPARAISON ENTRE SOM ET E-SOM

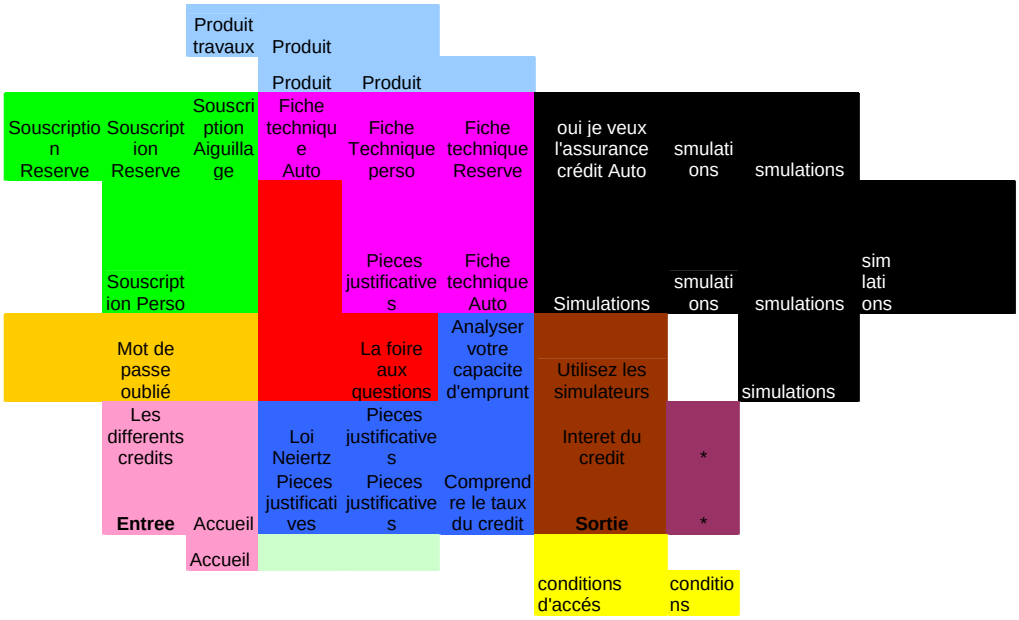


FIG 6.6. *CARTOGRAPHIE DU SITE (WWW.123CREDIT.COM) : APRES L'ARRIVEE DE 6 BASES*

6.6 Conclusion

Nous avons développé une idée qui consiste à traiter les nouvelles données qui arrivent, en utilisant le même modèle issu des données antérieures. En d'autres termes, nous avons rendu dynamique la création du modèle de la classification (la carte). Il s'agit donc, d'une carte qui change de figure dans le temps.

L'algorithme que nous avons proposé, respecte les mêmes règles de la création des cartes topologiques SOM (initialisation, compétition et adaptation) et permet de rendre la création des neurones de la carte de manière dynamique et dépendante de la nature des informations qui arrivent dans le temps.

Nous avons donc, rendu évolutive, la découverte de l'espace topologique : i.e. le nombre de regroupement homogènes (clusters)

change dans le temps en fonction des informations complémentaires qui arrivent. Une procédure tout à fait intéressante pour une mise à jour intelligente de la cartographie.

7 Conclusion générale et perspectives

Nous nous somme intéressé dans cette thèse à la modélisation comportementale des internautes face à des site web. Nous avons étudié principalement le comportement des modèles connexionnistes sur des formes basées sur des traces de navigation.

Nous avons d'abord abordé le problème délicat des données à utiliser pour la modélisation. Nous avons proposé plusieurs méthodes de codage pour formater les données basées sur des séquences afin de les utiliser dans des systèmes d'apprentissage automatique. Le processus de formatage que nous avons développé, nous a permis de préparer les données pour en optimiser l'exploitation selon la nature de l'analyse souhaitée et de la technique de Data Mining à mettre en œuvre.

Nous avons ensuite analysé les propriétés de l'algorithme d'apprentissage non supervisé des cartes topologiques de Kohonen (SOM). En apportant quelques modifications à cet algorithme, nous avons pu établir une cartographie d'un site web. Avec cette cartographie, nous avons pu montrer le plan du site tel que les internautes l'utilisent. Nous avons donc, établi un processus de visualisation qui réduit des grandes masses de données – impossible à comprendre a priori – en une simple projection bidimensionnelle, aussi bien simple que significative.

Ainsi, nous avons étudié l'impact de la programmation dynamique (DTW) sur la reconnaissance des sessions de navigation sur le site web. Cette technique nous a permis de reconnaître de quel comportement de navigation « type » une session quelconque est la plus proche. Les sessions traitées se présentaient sous la forme d'une séquence de « neurones » de la carte topologique qui a servi pour la visualisation.

D'autre part, la modélisation par mélange de chaînes de Markov a été appliquée à la classification des sessions de navigation, en prenant les neurones de la carte topologique pour des états de chaînes de Markov ; et le trafic dans la carte comme le système de transition dans le modèle Markovien. Pour l'algorithme d'apprentissage des paramètres de ce modèle (EM), une nouvelle méthode d'initialisation a été introduite. Le principe en était de calculer les paramètres initiaux à partir des classes obtenues par une classification ascendante hiérarchique (CAH) d'un échantillon de sessions de navigation, dont nous avons calculé la dissimilarité par la programmation dynamique.

De plus, confronté à l'arrivée dynamique des fichiers de navigation, nous avons proposé une version évolutive sur les cartes topologiques. Cette version a permis de créer les neurones de la carte topologique de manière dynamique et dépendante de la nature des informations qui arrivent dans le temps. Nous avons donc, rendu évolutive, la découverte de l'espace topologique : i.e. le nombre de regroupement homogènes (classes) change dans le temps en fonction des informations complémentaires qui arrivent. Une procédure tout à fait intéressante pour une mise à jour intelligente de la cartographie.

Une perspective consiste à étudier les variables caractérisant les traces de navigation des internautes. Il s'agit de trouver un sous ensemble de caractéristiques pertinentes pour le processus de modélisation du comportement de l'utilisateur. Cette sélection permettra d'une part, de réduire la dimension de l'espace de travail, et d'autre part d'optimiser la modélisation en ne gardant que l'information pertinente. Une dernière perspective consiste à faire coopérer des modèles connexionnistes pour la prédiction à court terme du comportement de l'utilisateur afin de le rediriger vers ses besoins prioritaires.

8 Brevets et Publications

BREVET NUMSIGHT: Procédé de traitement de fichiers de requêtes Internet N° : **WO02059781**

K. Benabdeslem, Y. Bennani, E. Janvier**. JT-EDI'01** : « Analyse comportementale des internautes », dans les actes de la conférence, France, Mars 2001

K. Benabdeslem, Y. Bennani, E. Janvier : **AICCSA'01** « Connectionnist approach for Website visitors behaviors mining ». In the proceedings of ACS/IEEE International Conference on Computer Systems and Applications, Lebanon, June 2001.

K. Benabdeslem, Y**.** Bennani, E. Janvier, **ICANN'02**. « Visualization and analysis of web navigation data ». LNCS2415 (Springer), pp 486-491, Spain, August 2002.

A. Zeboulon, Y. Bennani**, K. Benabdeslem. AICCSA'03**: « Hybrid Connectionnist Approach for Knowledge Discovery from Web Navigation Patterns » In the proceedings of ACS/IEEE International Conference on Computer Systems and Applications, Tunisia, July 2003.

Numéro de publication :**WO02059781**
Date de publication :**01.08.2002**

N° de dépôt : **WOFR0200303** Date de dépôt : **25.01.2002**

N° de priorité : **FR01/01013** Date de priorité : **25.01.2001**

Indice principal CIB : **G06F-017/30**

Déposant(s) : **NUMSIGHT SA**

Inventeur(s) : **JANVIER, Eric, BENABDESLEM, Khalid, BENNANI, Younes**

PROCEDE DE TRAITEMENT DE FICHIERS DE REQUETES INTERNET

L'invention concerne un procédé de traitement d'historiques de requêtes Internet adressées par un ensemble de clients à un serveur de site Internet, les clients et le serveur étant connectés au réseau Internet et le serveur comportant un certain nombre de pages. Ce procédé comprend des étapes consistant à: établir un historique de requêtes audit serveur; classer ces requêtes par groupes de clients; déduire des requêtes classées un historique des transactions effectuées successivement par chaque groupe de clients; établir pour chaque groupe de clients un état de chemins partiels centrés sur chaque page dudit serveur; déduire desdits état une représentation multidimensionnelle des transitions de page à page effectuées sur le serveur; réduire ladite représentation multidimensionnelle à une représentation à au plus trois dimensions; et établir, à partir de la représentation à au plus trois dimensions, une correspondance entre chaque page du site et une position dans cette représentation.

Etats désignés : **AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA CH CN CO CR CU CZ DE DK DM DZ EC EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NO NZ OM PH PL PT RO RU SD SE SG SI SK SL TJ TM TN TR TT TZ UA UG US UZ VN YU ZA ZM ZW GH GM KE LS MW MZ SD SL SZ TZ UG ZM ZW AM AZ BY KG KZ MD RU TJ TM AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE TR BF BJ CF CG CI CM GA GN GQ GW ML MR NE SN TD TG**

JT-EDI'01 : 1ère journée thématique d'exploration de données sur Internet.

La date : Vendredi 02 Mars 2001

Lieu : Université Paris 13 – Institut Galilée

Titre :

Analyse des données comportementales d'Internautes

Analyse des données comportementales d'Internautes

Khalid Benabdeslem [1,2] **Younes Bennani [1]** **Eric Janvier [2]**

[1]LIPN-CNRS, Université Paris 13
99 Av Jean Baptiste Clément 93430 Villetaneuse, France
{ kbe, younes} @lipn.univ-paris13.fr

[2]NumSight
Champ Montant 16 C 2074 Marin – Suisse
{k.benabdeslem, e.janvier}@numsight.com

Résumé

L'avènement du commerce électronique a révolutionné toutes les industries. Tout aspect commercial, à partir du point de vente jusqu'à la livraison finale a besoin d'être automatisé et disponible 24h sur 24h partout dans le monde. Cet article présente une formulation du problème de l'analyse des comportements d'Internautes, et propose un système hybride, Maxeem, basé sur la coopération des techniques d'apprentissage connexionnistes et des statistiques modernes. Étant une valide approche de modélisation d'utilisateurs, Maxeem a tendance à représenter une parfaite solution innovante de eMarketing permettant de mettre en place facilement une démarche Marketing professionnelle sur Internet.

1. Introduction

La prolifération de l'information sur le Web avait rendu nécessaire, la personnalisation de l'espace de cette information, cela veut dire que l'interaction de l'utilisateur avec cet espace est nécessairement basée sur des caractéristiques concernant son profil de comportement. La personnalisation peut être faite via des mécanismes de traitements d'informations (par exemple : les moteurs de recherche) ou bien faite d'une manière « end to end » en rendant le sites Web adaptatifs. Les travaux initiaux dans le domaine, ont été principalement focalisés sur la création des entités intermédiaires (systèmes à base de requêtes) souvent considérées comme des systèmes de recommandation. La personnalisation « end to end » est utilisée dans des sites Web adaptatifs, qui changent les informations retournées en réponses à des requêtes formulées par l'utilisateur. Plusieurs formes primitives de ce type de

personnalisation peuvent être vues dans des sites qui demandent aux utilisateurs de fournir des informations personnelles (adresse, numéro de téléphone, code zip, mots clé indiquant leurs intérêts). Ce type de personnalisation nécessite une acquisition explicite de données , qui sont utilisées ensuite dans un modèle de traitement de tâches – produites par l'utilisateur, sous formes de requêtes – afin de le comprendre, en d'autres termes pour extraire son profil. Cependant, ce type d'acquisition n'est pas satisfaisant, du fait que l'utilisateur fournit rarement ses informations personnelles via le Web. Pour cette raison, nous proposons de focaliser notre étude, dans un cadre d'adaptation d'interface (dans notre cas l'interface d'un site Web) pour extraire des profils d'internautes sans leurs demander de remplir des formulaires. Par conséquent, la découverte de profils d'internautes peut être considérée comme un processus de modélisation d'utilisateur. En effet, il peut être vu sous toutes les dimensions qui décrivent la modélisation ; il en représente donc une application particulière.
Le Web Mining[2] représente des caractéristiques qui rendent complexe son interprétation, cette complexité est due en grande partie au :

- Manque de données pertinentes (de nature numérique et symbolique)
- Variabilité inter et intra-Internaute qui explique le changement de comportements
- Acquisition implicite des données qui inclut beaucoup de bruits

Il est important de mentionner que plusieurs efforts ont été liés à des techniques relativement simples, qui peuvent être inadéquates au profil réel de l'utilisateur, du moment qu'elles soient incapables de traiter le bruit souvent trouvé dans les formes décrivant l'utilisateur [01]. Le Web Mining améliore les données qui sont légèrement infectées par le bruit. Des informations incomplètes peuvent facilement apparaître dans l'espace de données. Cela est dû à une large variété de raisons de connexion, d'ouverture de session et de navigation. De plus, le degré de contamination du bruit est rarement connu a priori.

Considérons une situation d'analyse des entités de Log[3] pour découvrir les formes d'accès aux informations d'un site. Il existe un pourcentage de temps (qui ne dépasse parfois pas les 30%) que l'utilisateur dépense pour visualiser les pages du site, il n'en suit donc aucune forme particulière.

[2] Web Mining : un terme souvent utilisé dans le domaine du Data Mining sur Internet, il concerne la fouille des données récupérées via le Web.
[3] Ce sont les données enregistrées par le serveur, elles représentent toutes les requêtes formulées par les utilisateurs d'un site, une explication plus détaillée sera donnée ultérieurement.

Par exemple, un utilisateur qui consulte un site de journal d'informations (comme TF1) pour la sport, peut aussi visiter la section politique dans une autre page de navigation.
Les caches Web causent aussi un autre problème pour toutes les techniques du Web Mining. Dans notre cas, nous prenons l'hypothèse qu'aucun cache n'est utilisé dans HTTP[4]. Ce problème peut donc être évité.

2. Modélisation d'utilisateur : formulation du problème et techniques de conception

Les recherches en Interaction Homme-Machine (IHM) et en Interface Adaptative (IA)[5] visent à rendre la machine facilement manipulable pour l'utilisateur. Cette facilité peut incorporer des notions d'intuition, de besoins et de plaisir [04].

Cependant, le challenge est clair : il consiste à concevoir des systèmes qui permettent aux utilisateurs de répondre à leurs besoins, cela nécessite a priori, plusieurs informations sur leurs caractéristiques afin de les modéliser pour pouvoir les comprendre, en d'autres termes, découvrir leurs profils suivant leurs interactions avec l'interface. Pour effectuer une bonne modélisation d'un utilisateur, nous avons besoin de bien le connaître, c'est à dire d'extraire des informations sur lui même pas en terme personnel mais en terme comportemental[6]. Pour cela, nous sommes amenés à étudier son interaction en construisant un système d'analyse de traces.

Les informations observées sur les caractéristiques d'utilisateurs peuvent être brûlées, par conséquent, tous les aspects d'amélioration de formes utilisateurs doivent être examinés avant la conception de tout système de prise de décision. En d'autres termes, les informations sur l'utilisateur peuvent être manipulées par un processus de reconnaissance des formes (ex : des formes comportementales, des formes de connaissance, …) basé sur le contexte de l'interaction. Dans ce sens, le système a pour objectif, d'établir des profils utilisateurs complets et consistants à partir d'informations incomplètes et bruitées. La modélisation d'utilisateur requiert donc, des caractéristiques d'association et de classification de formes. Les réseaux connexionnistes possèdent ces caractéristiques, et peuvent donc, être utilisés pour la découverte des modèles utilisateurs, en plus ils peuvent être entraînés à générer des

[4] Protocole de transfert d'hyper texte.
[5] IA : Abréviation pour Interface adaptative, elle est souvent utilisée pour l'intelligence artificielle.
[6] A vrai dire, un système de profiling nécessite tout types d'informations pouvant décrire un utilisateur, cependant un minimum d'information est requis. Cela dit, son comportement doit être étudié suivant sa façon d'interagir avec l'interface.

inférences. Les informations contenues dans les exemples sont utilisées pour construire une structure de décision qui pourra être utilisée afin de reconnaître de nouvelles formes. Dans ce cas, une forme est présentée à un système d'apprentissage connexionniste de la même façon que les exemples entraînés auparavant, et par conséquent le système retourne la classe d'appartenance de cette forme.
Traditionnellement, le modélisation d'utilisateur avait employé des techniques basées sur des connaissances. Ces techniques sont puissantes et efficaces mais souffrent de plusieurs inconvénients. En effet, le problème le plus persistant est celui de l'acquisition des connaissances : la formulation et le rassemblage des connaissances sont consommables au niveau du temps et provoquent éventuellement des erreurs [04]. Afin d'être efficace, la base de connaissances doit être correcte et complète. Si les informations sont corrompues, les inférences seront fausses et si les informations ne sont pas disponibles, les inférences ne peuvent même pas être faites. Dans le modélisation d'utilisateur, les comportements varient dramatiquement et involontairement, ce qui rend difficile l'assurance de la complétude et la correction. Les méthodes d'apprentissage numériques possèdent un nombre d'avantages sur les approches à base de connaissances :

Elles apprennent à partir d'exemples et ne nécessitent pas la formulation explicite de connaissances et de règles. Cela évite le problème d'acquisition des connaissances et permet aux experts d'identifier les cas représentatifs de chaque type.

Ces méthodes permettent une généralisation qui réduit la nécessité de la complétude dans la connaissance fournie. Bien qu'ils procèdent de différentes manières, tous les systèmes de reconnaissance de formes reconnaissent des formes non vue auparavant. Cela dit, l'utilisation d'un ensemble complet d'exemples n'assurent pas nécessairement la performance du système.

Les classifieurs de formes sont plus portables et plus extensibles. En effet, les exemple peuvent être ajoutés, ou bien tout le système peut être ré entraîné pour reconnaître de nouvelles caractéristiques. Ces techniques sont aussi flexibles d'être combinées pour fournir des profils sophistiqués composés de plusieurs bases de classification [04].

3. Le E-Mining[7]

Le E-Mining peut être considéré comme un processus de modélisation d'utilisateur. En effet, il peut être vu sous toutes les dimensions décrites en modélisation, il en représente donc, une application particulière.

- Les utilisateurs, dans ce cas, sont les internautes
- Le type de modélisation est dynamique (analyse de mouvement dans le temps)[8]
- Les interfaces sont adaptatives et organisées sous forme d'hiérarchie de pages
- Les formes sont des traces de pages formant des sessions
- L'acquisition est explicite par défaut (recherche par mots clés, remplissage de formulaire). Dans notre cas, elle est implicite car elle ne dépend que des clics.

On peut donc, utiliser un processus de reconnaissance des formes pour construire des modèles d'internautes. Ce processus est appelé : E-Mining.
Le concept du E-Mining s'appuie sur le constat, qu'il existe au sein de chaque entreprise , des informations cachées dans le gisement de données. Il permet, grâce à un certain nombre de techniques spécifiques, de faire apparaître des connaissances. L'exploration des données se fait à l'initiative du système. Son but est de remplir l'une des tâches suivantes : classification, estimation, prédiction, regroupement par similitudes, segmentation, description et dans une moindre mesure, l'optimisation. De nombreuses adaptations de ces techniques se font sur différents supports et génèrent un nouveau vocabulaire (Text Mining, Multimédia Mining, Web Mining,…) [03].

Le E-Mining peut nous renseigner sur la volonté des clients dans leurs interactions. Il peut nous assister à améliorer la marge de bénéfice pour les entreprises, en contrôle de gain sur leurs stocks et à la finn d'améliorer la forme et la hiérarchie de leurs sites. La meilleure assistance est celle qui garde en ligne des clients satisfaits grâce à l'extraction de leurs profils [02]. Les techniques citées auparavant, permettent de déceler des habitudes de comportements en fonction de la navigation des utilisateurs sur le serveur. Sur certains serveurs, le nombre de connexions quotidiennes peut atteindre des volumes très importants. Il est alors intéressant pour l'entreprise de pouvoir

[7] Le mining est souvent utilisé pour les données sous le terme de data mining, ce qui signifie littéralement forage de données et comme tout forage, son but est d'extraire un élément : c'est la connaissance.
[8] Dans le site commerciaux , cette analyse concerne les transactions en ligne et l'extraction de connaissance sur les clients

exploiter l'information générée (gratuitement), principalement à des fins commerciales, pour répondre à des questions sur les clients comme :

Séquencement : s'ils achètent le produit A, achèteront-ils le produit C ou X et quand ?

Segmentation : qu'est ce qui différencie mes clients fidèles des autres ?

Profiling : qui sont mes clients potentiels et comment je les gardes ?

Association : quel rapport y a t-il entre genre de visiteurs et ventes dans le site ?

Classification : comment pourrais – je reconnaître le proportion élevée d'achat ?

Clustering : quels attributs assurent le retour des visiteurs dans mon site ?

Optimisation : comment procéder pour changer le design du site pour maximiser les ventes en ligne ?

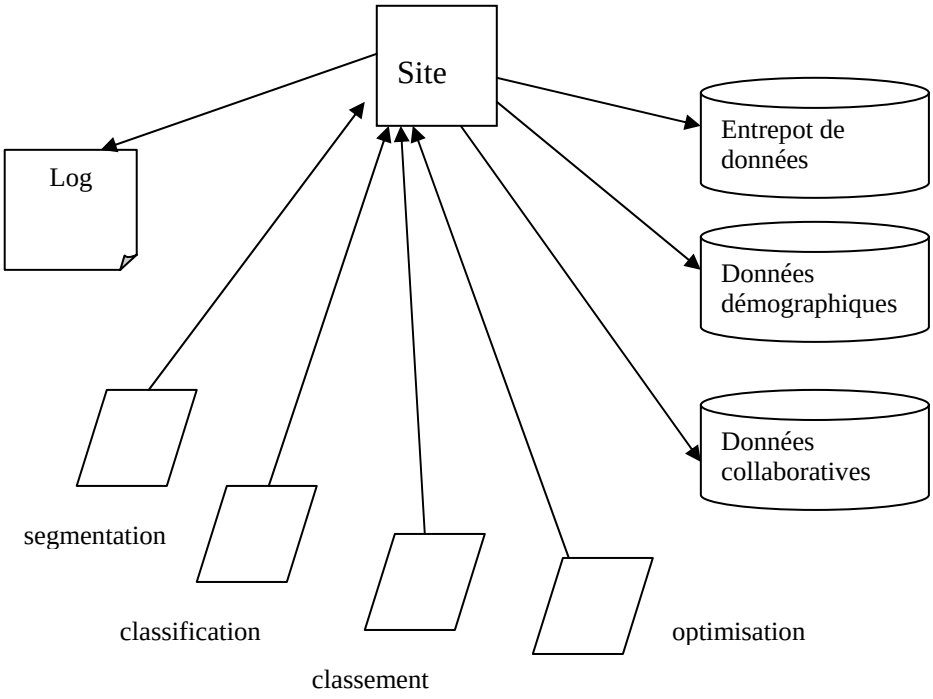


Figure1. Processus de E- Mining complet

4. La méthodologie utilisée en E-Mining

4.1. Acquisition des données

Le premier travail consiste à recueillir les données. Dans une analyse de type Data Mining, nous utilisons des grands volumes de données, principalement issues des fonctions de production, et stockées sous forme d'entrepôt de donnée[9]. Sur Internet, c'est l'utilisateur qui va fournir ces éléments par simple navigation :

- Adresse IP de l'utilisateur
- Date et heure de l'accès (et durée)
- Méthode utilisée (« GET », « POST »,…)
- Page accédée (adresse URI)
- Protocole de transmission (généralement http)
- Octets envoyés

4.2.Nettoyage des données

Comme dans tout travail de Data Mining, il faut nettoyer les données. Pour cela, nous supprimerons toutes les données portant une information qui ne sera pas prise en compte. Ce sera souvent le cas pour les fichiers image (extensions GIF, JPEG , JPG,…).

4.3. Formatage des données

Le formatage permet de préparer les données pour en optimiser l'exploitation, selon la nature de l'analyse souhaitée et la technique de Data Mining à mettre en œuvre.

4.4 Analyses liées au Data Mining

4.4.1. Analyse de chemins

L'objectif de cette analyse est de déterminer comment les utilisateurs parcourent le site.

[9] connu souvent sous le nom de Data Warhouse

Exemple de résultats :

- 85% des utilisateurs ont consulté 4 pages ou moins.
- 80% des connexions arrivent sur /societe /produits.
- 75% des clients qui ont accéder à la page /societe /offre-speciale.html venaient de la page d 'accueil.

Le premier résultat montre qu'il faut adapter l'arborescence (ou les liens transversaux) pour que les pages intéressantes se trouvent dans les quatre premières pages.

Le second résultat montre que la majorité des utilisateurs ne passent pas par la page racine (supposée être /societe dans ce cas). Il faudrait alors peut-être prévoir un lien vers celle-ci.

Le troisième résultat montre que l'accroche de la page d'accueil sur la page offre-speciale est très efficace. Peut-être faudrait-il l'utiliser sur d'autres pages ?

4.4.2. Les règles d'associations

L'objectif de cette analyse est de déterminer quelles pages sont souvent associées (même sans lien hypertexte) ou l'origine d'une action (achat en ligne par exemple).
Exemples :

30% des clients qui ont accédé à la page /societe/offre-speciale.html ont fait un achat en ligne sur la page/societe/produits/produit1

40% des clients qui ont accédé à la page /societe/produits/produit1.html ont également consulté la page /societe/produits/produit5

L'analyse de ces résultats va permettre de créer de nouveaux liens ou d'expliquer les actions (l'achat en ligne par exemple).

4.4.3. Les règles de classification

L'objet de cette analyse est de regrouper les individus en fonction de leurs actions et de renseignements propres aux personnes. L'intérêt est de former des groupes de personnes ayant des comportements similaires. Dans un premier temps, à des fin d'analyse (qui sont mes visiteurs ? mes clients ?) et dans un second temps à des fins d'anticipation (proposer le produit le plus adapté à un visiteur).

5. Maxeem : une application intelligente de E-Mining

Malgré l'absence de plusieurs connaissances a priori sur toutes les formes possibles, nous avons élaboré une solution de Data Mining composée d'un mélange de techniques de statistiques et d'intelligence artificielle. Cette solution est représentée par un produit développé par la société Numsight S.A. sous le nom commercial : **Maxeem.** Ce dernier permet de mettre en place facilement une démarche Marketing professionnelle sur Internet.

Maxeem a comme vocation de :

- Fournir aux marketeurs des sites Internet un outil dynamique de compréhension et de pilotage de leurs sites pour en augmenter la pertinence marketing.
- Permettre le stockage des sessions historiques sur un espace disque très limité, peu coûteux et simple d'utilisation.
- Augmenter le taux de conversion des entrants en clients / consommateurs en mettant en place une personnalisation de l'offre en temps réel basé sur les besoins instantanés de l'internaute.

5.1. La réponse de Maxeem aux problématiques liées aux sites

Maxeem apporte la réponse aux questions qui se pose tout site Internet :

Comment fonctionne mon site ?
- Quel est le plan réel de mon site tel que les internautes l'utilisent ?
- Comment les visiteurs circulent-ils dans mon site ? Quels chemins suivent-ils ?
- Comment expliquer la réussite ou l'échec des sessions ?
- Comment puis-je améliorer l'attractivité de mon site ?
- Quels parcours dois-je proposer aux visiteurs ?

Comment puis-je piloter la pertinence marketing de mon site ?
- Comment suivre et mesurer la pertinence de mon site ?
- Comment le faire évoluer dans une approche progressive ?
- Comment me construire une base d'historiques simple et opérationnelle ?

Quels sont les visiteurs qui vont avoir besoin d'aide ?
- Qui sont les visiteurs qui vont finalement acheter ou consommer mes services ?
- Puis-je identifier ceux qui vont sortir sans acheter ou consommer ?

- Puis-je savoir combien de temps ces derniers vont rester sur mon site ?

Que puis-je faire pour influencer le visiteur hésitant et en faire un client ?

- Quels sont les besoins de ces visiteurs hésitants ?
- Que dois-je proposer pour répondre à leur besoins et en faire des clients satisfaits ?

5.2. La suite logicielle Maxeem

Une suite unique de logiciels Internet intelligents et auto-adaptatifs agissant en temps réel pour améliorer la relation avec les internautes clients/prospects ; et l'efficacité des actions marketing.

5.2.1. Le module MaxeeMap

- D'identifier les chemins critiques effectivement suivis par les internautes, explicatifs des succès ou échecs des sessions en regard des objectifs du site (Achat, panier moyen, consommation de services, durée de session, …)

- De disposer en continue d'une représentation cartographique simple de son site. Ce plan du site représente en deux dimensions les chemins parcourus et les centres d'intérêts des internautes. Cette représentation peut être déclinée selon toutes les typologies pertinentes choisies par le responsable Marketing du site (Générale, jour, tranches horaires, type de client, …)

- De tester On-Line de nouveaux dispositifs marketing sur le site (structure du site, visuels, contenu, offre produit ou service, …) et d'en mesurer rapidement les impacts sur le comportement des internautes.

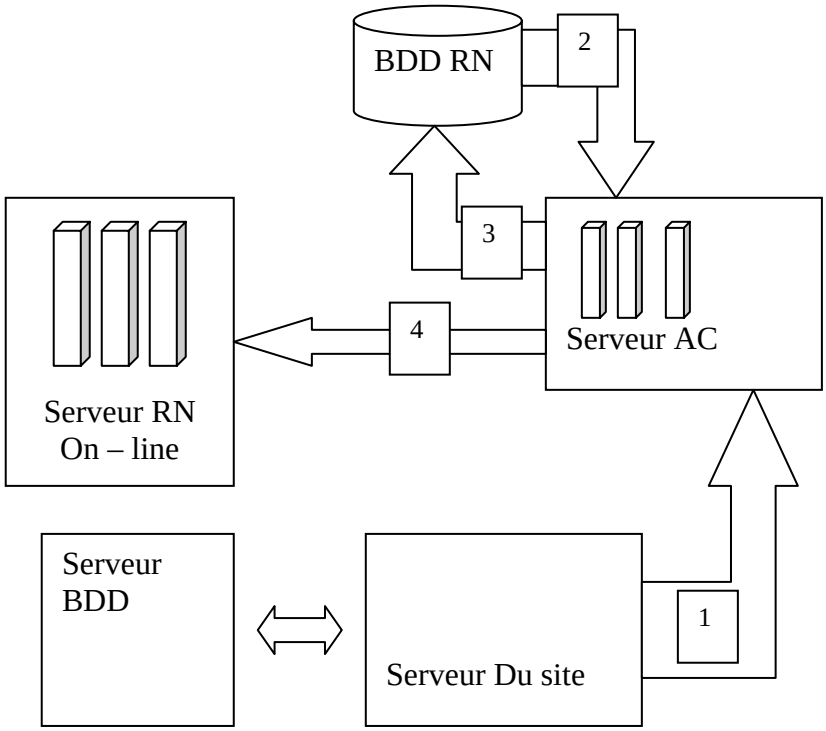


Figure 2. Architecture générale de fonctionnement de MaxeemMap.

1. comme le site ne s'arrête pas de changer, « le serveur Analyse Comportementale » est paramétré pour analyser les flux http générés par le serveur du site.

2. la base de données de profils utilisateurs est analysée par la suite, utilisant un modèle de prédiction à base de réseaux de neurones (RN) pour trouver les règles qui prédisent quels utilisateurs sont sensibles de devenir des bons clients, quel click serait le plus important pour chaque utilisateur. Typiquement, l'objectif primaire est de générer des modèles qui lient les ressources avec l'utilisateur.

3. les profils ressources et les profils utilisateurs sont stockés dans une base de données pour être exploités ultérieurement.

4. le serveur de l'analyse comportementale envoie les modèles mis à jour au serveur On-Line : les modèles contiennent les règles et formes que le serveur On-Line utilisera pour lier « rapidement » l'utilisateur à centre d'intérêt.

5.2.2. Le module MaxeemInfluence

MaxeemInfluence est composé de deux sous-modules omplémentaires permettant de mettre en œuvre une personnalisation en ligne adaptée à chaque internaute :

Module de prédiction permettant, à partir de l'observation des déplacements de l'internaute, de déterminer le plus rapidement possible :

- la durée probable de la session,
- la probabilité de conversion d'un visiteur en consommateur en fonction des objectifs que s'est assigné le site (achat de produits ou services, contenu, audience, durée de session).

Module d'identification du besoin prioritaire de l'internaute permettant l'envoi de codes action et en temps réel le déclenchement sur la plate-forme du site d'actions marketing personnalisé et adapté au besoin de chaque internaute.

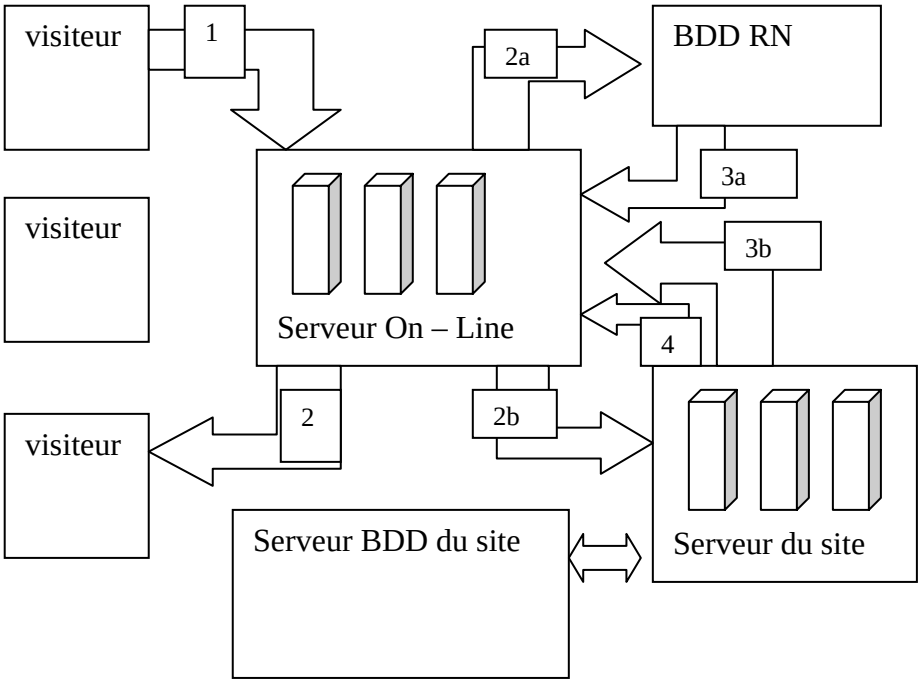


Figure 3. Architecture générale du Profiling On - Line

1. Des internautes visitent le site commercial. Le serveur RN On-Line intercepte les pages

2a. Maxeem exécute la collection des données et met à jour le profil utilisateur stocké dans la base de donnée RN. Ce profil utilisateur contient les préférences et les intérêts de l'utilisateur.

2b. Maxeem transfère les pages requises au serveur du site.

3a. Maxeem va chercher le profil utilisateur couvrant le profil ressource à partir de la base de données RN.

3b. Maxeem reçoit le contenu des pages à partir du serveur du site. Il applique ensuite le modèle du Data Mining au profil utilisateur.

4. Basés sur les résultats de cette analyse, des contenus personnalisés sont recherchés dans le serveur du site (cela peut être des bannières publicitaires, des liens additionnels, ou des contenus dynamiques).

5. Le serveur Maxeem On-Line envoie les pages assemblées au poste client.

5.2.3. Le module Data Webhouse

Grâce à ce module, **Maxeem** permet :

Le stockage de toutes les sessions sur des espaces disques très faible capacité :

- L'algorithme de compression (facteur de 10.000) des signaux http permet de stocker les historiques annuels de visites de très gros sites Internet sur quelques dizaines de Méga Octets.

La mise à disposition d'une base de données sur les comportements et la performance souple et facile d'utilisation.

- Traitement sur PC avec la plupart des logiciels statistiques du marché,
- Dépendance réduite vis à vis des informaticiens,
- Accès à un marketing personnalisé à grande échelle.

La réduction des investissements

- Mise en place immédiate et simple,
- Pas de DataMart[10] spécifique coûteux,
- Interface simple avec les bases de données clients existantes.

[10] base de données orientée sujet mise à la disposition des utilisateurs dans un contexte décisionnel décentralisé

6. Conclusion

Il faut savoir que les techniques d'analyse en Data Mining ne sont pas exclusivement attachées entre elles et n'ont pas besoin d'être utilisées séquentiellement, ce sont les besoins d'application qui déterminent quand chacune d'elles doit être utilisée. Similairement à la modélisation d'utilisateur, les outils en Data Mining sont basés sur les techniques d'intelligences artificielle désignées à simuler la perception et l'apprentissage humains. L'avantage des outils de Data Mining réside dans la façon de traiter les formes de manière autonome, elles sont conduites par des données contenues dans les fichiers Log et les bases de données générées par le Web plutôt que de fournir des hypothèses personnelles sur les comportements. Dans ce concept, Maxeem représente un outil d'amélioration de la performance des sites Web visant à instaurer une véritable relation « win-win » entre le site et ses visiteurs. Sur Internet comme ailleurs le « client est roi ». Être performant suppose de comprendre le client et de le respecter. **Maxeem** le traite comme tel : Confidentialité (respect absolu de l'internaute et de sa vie privée, pas d'espionnage, pas de « cookies » violant l'intimité de l'internaute, historiques stockés de manière anonyme), Confort (respect du confort et de la navigation de l'internaute, pas de questions en ligne venant perturber sa session), Discrétion (Maxeem est basé sur l'observation, assistance en temps réel transparente pour l'internaute, aucune altération des performances techniques du site).

7. Références

[01] A. Joshi – On Mining Web Access Logs. Technical Report, CS Departement, University of Maryland, College Park, 1999.

[02] J.Mean – Data Mining your WebSite. Digital Press, Butterworth, 1999.

[03] B. Mobasher - Web Mining: Pattern Discovery from World Wide Web Transactions A Research Overview. Présentation par B. Mobasher du projet Webminer du laboratoire de recherche de l'université du Minnesota, juillet 1997, http:// www - users.cs.umn.edu/~mobasher/webminer.html.

[04] J. Filnay and R.Beale – Pattern Recognition and Classification in Dynamic and Static User Modeling. In Neural Networks and Pattern Recognition Human – Computer Interaction, R. Beale et J. Finlay éditeurs, Elis Horwood, 1992.

AICCSA'01 : ACS/IEEE International Conference on Computer Systems and Applications

La date : 25 – 29 Juin 2001

Lieu : Lebanese American University

Titre :

Connectionnist approach for Website visitors behaviors mining

Connectionist Approach for Website Visitors Behaviors Mining

Khalid Benabdeslem [1,2] **Younès Bennani** [1] **Eric Janvier** [2]

[1] **LIPN – CNRS, Université Paris 13**
99 Av Jean Baptiste Clément 93430 Villetaneuse, France
{kbe, younes}@lipn.univ-paris13.fr

[2] **NumSight**
Champ Montant 16 C 2074 Marin – Suisse
{k.benabdeslem, e.janvier}@numsight.com

Abstract

In this paper, we propose a new version of the topological maps algorithm, which has been used to cluster web site visitors. These are characterized by partially redundant variables over time. In this version, we only consider the input vectors neurons that participate in the selection of the winning neuron in the map. In order to identify these neurons, we use a binary function. Subsequently, we apply a partial modification on the weights that relate them to the winning neuron.
Using this new version, we obtained a clustering of web site visitors behavior, which has been difficult to analyze before.
This clustering will allow a recommendation system to satisfy the web site visitor needs based on his cluster membership at each step in time.

1. Introduction

Websites of all sizes are currently experiencing an explosive growth in their capabilities to generate massive amounts of server data. However, there are not able to understand the customer whose generate these data by their requests. Thus, they necessitate a technic of E-mining. It is an iterative process of analyzing the patterns of the online transactions and extracting knowledge about the activities of website visitors. In this context, we define a notion of user sessions, such as a compact temporal sequences of access pages of the site. We propose to use a relational clustering in order to cluster similar pages before doing the behavior clustering and the movement analysis.

Considering the user's traces such sessions, we propose to extract their requests in a temporal window saved in the server.
Finally, we do some modifications in the topological map algorithm for extracting partial behaviors clusters, which allows us to understand their particulars needs in the site.

2. User modeling

Behavior extraction of web site visitors can be considered as user modeling process. In fact, it can be viewed over all dimensions, which describe modeling; thus, it represents a particular application [07].

The observed information about a user's characteristics may be mixed with noises or inconsistencies. Therefore, all aspects of user's performance patterns must be examined before any system decision can be made. In other words, the information about a user should be processed by pattern recognition so that the system can establish complete and consistent user profiles. In this sense, user modeling is a process of recognizing a user's patterns (eg. a user's behavior pattern, knowledge pattern, cognitive pattern, etc.) based on the context on interaction [08].

As a pattern recognition process, user modeling requires the feature or pattern association and classification, Neural networks have all these features and therefore can be used for implementing user models. In addition, neural networks can be trained to generalize inferences.

E–mining can be considered as a user modeling process. In fact, it can be viewed over all dimensions described on modeling, it therefore represents a particular application.
Users: web site visitors
Type of modeling: dynamic, movement analysis in time.
Interface: adaptive organized such as tree of pages.
Patterns: sessions of pages
Acquisition: explicit in default (keyword research, formulas filling). In our case, it is implicit because it depends only on clicks.

Data exploration is done on a system initiative. His goal is to process one of following tasks: classification, estimation, prediction, clustering and optimization [05].

3. Connectionist unsupervised learning

There are a lot of problems in which we dispose a set of data $A=\{z_1, z_2, \ldots, z_N\}$, without any label attached to *z*. however, we hope to extract the pertinent information of such a set.
Because we don't associate any response known a priori to the input pattern, it can not have supervision by the user in learning, so we call our learning unsupervised.
In this context, the algorithm of topological maps proposed by Kohonen [09] is used for data organized on tables: patterns / variables, where the variables are fixed for all examples on process. During learning, the neurons of entered vectors are attached to values of defined variables.
In our case, a pattern is characterized by *p* variables, is not directly presented to the network. The set of these variables is presented on succession of slices, which are redundantly presented in time. This explains the behavior change if it exists. Thus, the input code vector will be dynamic.
We proposed to process a new version on weight modifications in the algorithm of Kohonen by adding an *election* function. It is a binary function, which allows masking the neurons, which don't participate in the election of the map's neuron.

We present an example of data coding:
Let *zi* a pattern characterized by *p* variables
The individual's pattern is presented such as discrete time session. We select partial characteristics at each temporal interval. These characteristics are presented to the network and they consider just their associated weights.

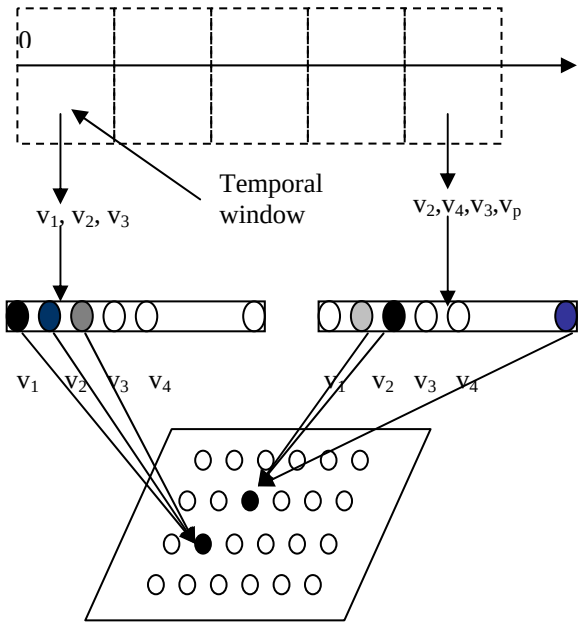


Figure 1. Topological map with map's neurons election depending on dynamic coding.

At each temporal interval, we extract just a subset of our pattern's characteristics, we select the neurons associated at these characteristics, and then we present the window's content to the Kohonen´s map. We consider just the entries neurons, which participate in the election (Figure 1).

We can remark a difference in colors of neurons of the same window, it means that there are a difference in values of variables in data. Particularly, in our application that explicit the user interest in each page during his web site navigation.

Concretely, each pattern represents a set of temporal windows. Each one contains a subset of variables. Each window provides its variables to the Kohonen map in order to select a neuron. In the end of session, the pattern will be represented by a set of winner neurons in the map.

Therefore, we will have a clustering of pattern *slices*, that means that it can be within class *i* according to a given subset of variables and within class *j* according to an other subset of variables. For our application, this stimulus describes explicitly the behavior change which can have a web site visitor during his navigation. If the pattern appears in a lot of clusters, we can calculate the distance between these classes in order to evaluate his behavior change.

4. Data base

The access log for a given web server consists of a record of all files accessed by user. His traces are saved in a log file. First, we filter out log entries that are not germane for our task (e.g. images, failure download).

After the filtering, we try to extract the pages, to calculate the consulting times and to define a sessions.

4.1. Extraction and coding

In our case, we study the phenomena of navigation by doing an analysis and clustering of the movements in the site.We want to recode the user interest at each step in time.

The aim is to caricature the site by a topological map which groups the same behaviors and which codes the user movement optimally.
In the log file, the most important field, is the visual page at a given instant. Subsequently, each page will be a characteristic of user interest in this instant.
The page is characterized by two symbolic variables: address and content. We define their similarities as follow:

Address URL similarity

$$S_{Address}(x, y) = 1 - \frac{f(x,g(x,y)) + f(y,g(x,y))}{f(x,racine) + f(y,racine)}$$

Where *f* gives the number of links between two nodes (*x, y*) and *g* gives the most specific common abstraction of two nodes.

Content page similarity

The page content is described by HTML language format.
So that, we define the similarity between two pages contents by taking their common keywords:

$$S_{Content}(x, y) = \begin{cases} 0 \text{ if } x \cap y = \phi \\ \dfrac{Card(x \cap y)}{card(x \cup y)} \text{ otherwise} \end{cases}$$

After defining these two similarity measures, we weight them with a vector of weights ($w_{Addresss}$, $w_{Content}$) in order to finally compute the similarity between two pages (*x, y*) in the site:

$$S_{(Address,\ Content)}(x,y) = w_{Address}\, S_{Address} + w_{Content}\, S_{Content}$$

Therefore, we can define a similarity matrix between pages. We assume that the visited pages are symbolically presented in the file of data, we propose to do a relational clustering for optimizing the number of pages. For that, we use a hierarchical clustering which allow us to obtain clusters containing similar pages. Practically, this study allows us to reduce the number of neurons of input vectors.

4.2. Time dimension

The time between two requests includes the consulting time and the downloaded one (i.e. the time of the transfer in network), The consulting time is the time passed between the beginning of displaying a page and the beginning of displaying a followed one. In some applications [06], the download time is removed. They consider just the consulting time of the page. However, in our case, we assume that the download time represents the waiting of the user to get his request, having said that, we consider that this time represents his patience degree. Thus, the user interest in a page, is coded by the consulting time and the waiting one.

4.3. Sessionizing access log data

Our table of data contains a lot of sessions of many web site users. A session is defined as one temporal sequence of access pages of the site. In one area, the server provides rarely the user's identifier and in the other one, our application concerns the study of behaviors. It's for that reason, we define a user session as an access address IP providing that the time between two consecutive pages don't go on for more than a fixed threshold.

Each URL in the site is assigned by one index $j \in \{1, \ldots\ldots, N\}$
where N represents the total number of pages. These pages are reduced at *p* clusters containing similar pages.
Globally, a session can be coded as follow:

$$S_j^{(i)} = \begin{cases} 1 & \text{if user asks the } j^{ème} \text{ cluster of URLs during the } i^{ème} \text{ session} \\ 0 & \text{otherwise} \end{cases}$$

However, this coding don't consider the movement in the site and don't explicit the user interest over the pages.

4.4. Movement coding

The principle consists to shift a temporal window over the sessions in the site and it considers just the clusters of the pages included in this window. These clusters are characterized by an index, which selects the neurons, which participate in election, the others are inhibited in a given temporal interval.

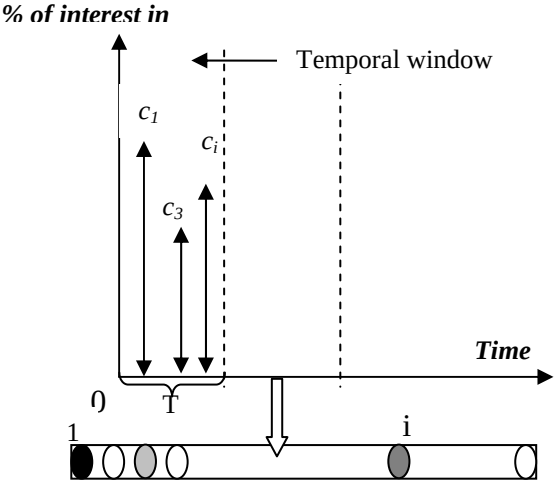


Figure 2. Dynamic coding of user interest in the site.

At each interval (T) the user divides this period over a number of clusters of pages c_i. The figure 3 show that the website visitor
In the figure, there is more interested by the first page cluster than the 3rd one.
The activation of each neuron is given as follow:

$$a_i = \frac{\sum_{j=1}^{N} t_j(page \in classe_i)}{T}$$, with T: temporal window length, N : number of visits of $cluster_i$, $t_j(page \in class_i)$: the jth period passed in one page included in $cluster_i$.

These components represent a percentage of user interest in each page at each temporal interval. In other words, we dynamically code our input vector, by computing the temporal window distribution over his visited pages.

5. Results

5.1. Behaviors Clustering

For user behaviors clustering, we use the Kohonen's algorithm with a new version over the weight modification presented in section 3.
The learning algorithm can be described as follow:

- T=0, we randomly initialize the weights W (codebook vectors)
- At time *t*, one sample vector z is randomly chosen from the input data set
- Distances (similarities) between the z and all the codebook vectors are computed
- The best matching unit (BMU), denoted here by *c* is the map unit whose weight vector is chosen to the *z*.

$$\|z - Wc\| = \min_i \{\|z - Wi\|\}$$

-
- The weight vectors are updated.
  The BMU and its topological neighbors are moved closer to the input vector in the input space.
  The update rule for the weight vector of unit *i* is:

$$w_i(t+1) = \begin{cases} w_i(t) + \varepsilon(t)\,[e(t).z(t) - w_i(t)] \;,\; i \in N_c(t) \\ w_i(t) \qquad\qquad\qquad\qquad\quad i \notin N_c(t) \end{cases}$$

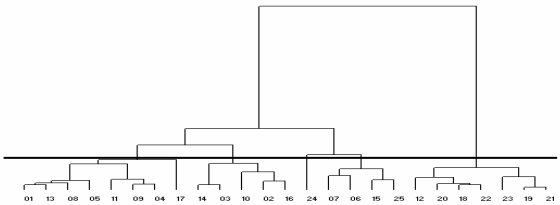
Where *t* denotes time, $N_c(t)$ is a non – increasing neighborhood function around the winner unit c, $0<\varepsilon(t)<1$ is a having coefficient, which is a decreasing function of time and e(t) is the election function defined as follow:

$$\text{With } e(i) = \begin{cases} 1 \;\; if\; i \in temporal\ window \\ 0 \;\; \text{otherwise} \end{cases}$$

The principle: the winner neuron in the map depends only of the neurons, which participate in its election.

Concretely, at each temporal interval, we code our neurons vector (classes of pages) included this interval. In other words, a sample is characterized by *k* variables, however at each step in time; there is only a part of these variables that characterize his behavior.

The previous algorithm allows us to cluster a partial user's behaviors interested by the same types of products (or services) included a similar pages.
This clustering allows in first time, to understand the needs of visitors (by labeling each class by one service or one set of products), and in second time, to provide to a recommending system the possibility of proposing products adapted to the user's needs.
We carry out a hierarchical clustering of the topological codebook vectors given in section 5.1.
Figure 3 gives the results of this second clustering. The labeling was performed using expert knowledge.



a: Entry zone
b : Purchase zone
c : Insignificant zone
d : Promotion products
e : Zone of hesitationBetween products
f: Zone of services : contact, games and ask for license.

Figure 3. Segmentation and labeling

5.2. Trajectories analysis

If we project a user's trace in a commercial web site over a topological map then we remark a similar behavior change and more significant one
Let the following description:

- T_1: he starts by consultation of contact zone.
- T_2: he goes to the purchase zone where he put products within a caddie.
- T_3: he returns to the home page where he searches an other products.
- T_4 : he is found in the insignificant zone (this zone means that, during the learning the coding system has selected just one neuron (in the temporal window)
- T_5: at this instant, he is found in the *dark* zone, it is a zone of hesitation. In fact, he was comparing between a lot of products.
- T_6: at this instant, he's in the purchase class again.
- T_7: he doesn't change his behavior. In fact, he bye (the curve in figure 4).

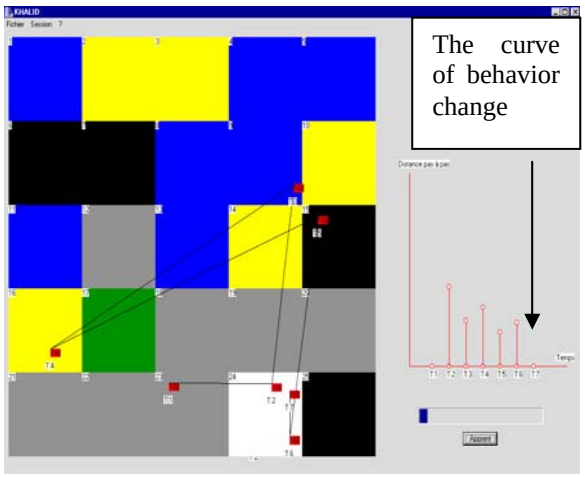


Figure 4. Representation of user's trajectory on the map

6. Conclusion

This work allowed us to analyze the proprieties of the topological map's learning algorithm. With modifying this algorithm, we could cluster web site visitors, which are characterized by slices of variables in time. In other words, in our application, we have found another aspect for characterizing the individuals. They are not known according to their characteristics fixed a priori. However they are represented as a sessions of variables, which are redundantly presented in time.

This formalism allowed us to cluster partial behaviors of web site visitors, and to caricature their movements in the site with a topological map. In addition, this map allows to understand the visitors needs (represented by products and services in each page of the site).

We have used a hierarchical clustering in order to optimize the number of classes. Concretely, that allows to a recommending system to cluster some of products (or services) according to similar needs.

Acknowledgements: this work was done in NumSight S.A, with the help of a team of customer's behavior modeling, that we thank for their collaborations. The authors would like to acknowledge Wstore (**www.wstore.com**) for provide us the data (Log Files) in order to test our modeling over their web site visitors.

7. References

[1] k. Benabdeslem and Y. Bennani, « Classification connexionniste pour l'analyse des comportements d'internautes », *Technical report, LIPN – CNRS*, France, 2000.

[2] L. Chittaro and R. Ranon, "Adding Adaptive Features to Virtual Reality Interfaces For E-Commerce". *In Proceeding International Conference On Adaptive Hypermedia and Adaptive Web Based Systems (AH2000), Lecture Notes in Computer Science, Springer*, Berlin, 2000.

[3] K. Abe, "Concept – Based Search and User Modeling in Information Retrieval Based on Human – Machine Dialogue". *In RIAO2000, Conference on "Content-Based Multimedia Information Access"*, Japan, 2000.

[4] A. Joshi, "On Mining Web Access Logs". *Technical Report, CS Department, University of Maryland, College Park*, 1999.

[5] J. Mena "Data Mining your Website". *Digital Press, Butterworth ,Heinemann*, 1999.

[6] M. Jaczynski and B. Trousse, "WWW Assisted Browsing by Reusing Past Navigations of a Group of Users". *In Advanced in Case-based Reaosning, 4th European Workshop on Case-Based Reasoning , Lecture Notes in Artificial Intelligence, 1488, pp160-171*, 1998.

[7] M. Dimitrova and al, "Neural Networks for Classification and Recognition of Individual Users in Adaptive Human Computer Interaction". *In Proceeding of the 12th IEEE International Symposium on intelligent control, pp101, 106*, 1997.

[8] T. Kohonen, "Self Organizing Map and Association Memories". *Springer, Vol 8, Springer Verlag*, 1984.

ICANN'02 : International Conference on Artificial Neural Networks

La date : 27 – 30 Août 2002

Lieu : University of Madrid

Titre :

Visualization and Navigation of Web Navigation Data

Visualization and Analysis of Web Navigation Data

Khalid Benabdeslem[1], Younes Bennani[1], Eric Janvier[2]

[1]LIPN – CNRS, University of Paris 13
99 Av Jean Baptiste Clément - 93430 Villetaneuse, France
{kbe, younes}@lipn-univ.paris13.fr
[2] NumSight, Champ Montant 16 C 2074 Marin – Switzerland
e.janvier@numsight.com

Abstract. In this paper, we present two new approaches for the analysis of web site users behaviors. The first one is a synthetic visualization of Log file data and the second one is a coding of sequence based data. This coding allows us to carry out a vector quantization, and thus to find meaningful prototypes of the data set. For this, first the set of sessions is partitioned and then a prototype is extracted from each of the resulting classes. This analytic process allows us to categorize the different web site users behaviors interested by a set of categories of pages in a commercial site.

1 Introduction

Clustering represents a fundamental and pratical methodology for the comprehension of great sets of data. For multivariate data, we can use geometrical distance concepts in an euclidian space, but for data consisting of sequences (or images for example), there is no geometric equivalent [5].
Traditionally, two general techniques are used to deal with this problem. The first approach is to reduce all heterogeneous data to one fixed vectorial representation. For example, vectorial space representation is largely used in information extraction for the representation of textual documents having different formats and different length and histograms are used to resume web access sequences. Practically, this technique is useful and has the advantage of being relatively simple. It makes it possible to profit from clustering algorithms advantages in vector space. One second approach of clustering is to use a defined distance between each pair of individuals. Given N objects, we generate a N^2 distances matrix. There is a significant number of clustering algorithms which can use this matrix (ex: AHC). However, for a great data base, time and space complexity becomes increasingly delicate for such algorithms. In this paper, we choose the first technique by using one coding method that we propose to reorganize sequence based data.

2 Data base

We would like to describe a method of exploratory analysis of dynamic behaviors for web site visitors. To validate our work, we use the access log representing thousands of individuals in a commercial web site.
Therefore, our data base includes individuals which represent web site visitors, in which each one is supposed to be interested by a set of pages. Each user (session) will be represented by a succession of pages[4].

Session N°	pages
S_1	2 7 4 1 5
S_2	8 4 32
S_3	3 9 23 20

Table. 1. Format of the navigation file.

The different numbers in the box "pages" each represent a page of the site.

3 Coding

Actually, the sites are dynamic. That means that the pages are not characterized by a fixed variables as: hierarchical address (URL), the content ....etc. They are represented by a numerical identifiers which have no meaning. For that reason, we have elaborated a novel method of coding sessions from the Log file which consists in characterizing a page by its passage importance, i.e. by its weight of precedence and succession relative to the other pages of the site which appear in the log file.
The principle is to calculate for each page its frequency of precedence and succession over all the other pages and to regroup these frequencies in one matrix : Pages × (previous Pages + next Pages) that we call : quasi-behavioral matrix. We remarked that this coding has a serious problem of size. In fact, if we don't have a enough pages in the site, we can not have a table with enough data to get a robust model. To solve this problem, we proposed to slide the matrix along the log file. In other words, we computed a quasi-behavioral matrix for each time period. This method allowed us to increase the number of samples and to have several examples for each page.

Example:
Let's suppose we have a Log file with 40 URLs over 30 days.
If we slide the matrix along the month day by day, we get a 40 × 30, ie 1200, samples base. We not only characterize the pages by behavioral variables i.e. as the web site visitors perceive the site, but we increase the information about the pages by coding their behavioral characteristics in detail, day by day.
For example, for URL_i , we can code its vector over all the others URLs as follow:
In the k^{th} day of the file, the URL_i appeared *a* times as an entry page, preceded *b* times by the URL_j, succeeded *c* times by URL_l and appeared *d* times as an exit page.

4 Neural clustering for the quasi – behavioral visualization

We remember that the log files have an important size and contains relevant information about the web site visitors behaviors.
For the profiles extraction, we use Kohonen's topological map algorithm [8] in order to group the similar pages[8], then we apply an ascendant hierarchical clustering to optimize the classes. Therefore, we talk about clusters of neurons.
The obtained clusters need to be labeled. However, at the time of use of the map, each URL can activate more than one neuron, depending on its passage importance change day after day. Theoretically, if the map is well done, the different neurons which are activated by the same page must be close to one another.
Practically, to label the map in this case, we proceed by a majority vote. We have applied this process to a log file of a commercial web site (www.123credit.com) having a size of 1 Giga byte, and got the following map:

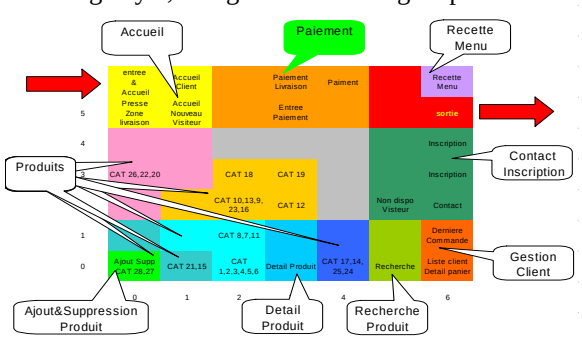


Fig. 1. Mapping of a commercial site (www.123credit.com)

5 Sessions coding

Instead of representing the sessions by the succession of visited pages, we represent them by the succession of the class of pages resulting from the unsupervised clustering. This coding reduces the variety of the traffic but it doesn't solve the problem of variability of the length. In fact, several problems appear in the treatment of sequence based data :
The presence of « holes » (missing data)
The complex variability of the length
The symbolic nature of the problem which implies a particular study about the similarity of the individuals. In order to solve this problem, we have developed a new method which allows to code the set of sessions in a table of individuals versus variables, whose number of columns is fixed and without any missing data.
The principle of this coding, that we called "position weight" consists in awarding a weight to each category according to its position in the session : the further a category appears in a session, the more its weight increases. By the way, the coding

also includes the frequency of the category in the session. The algorithmic form of this coding is the following :

Coding of "position weight" algorithm

```
// Initialization
For i = 1 to N do
 Weight(i) = 0
 Frequency(i)=0
End for
// position weight and  frequencies
For j ∈ session do
 If  j = 1st category of the session then
  Weight(j) = α //α : is an initial parameter
 End if
 If  Frequency(i) = 0 then
  Weight(j) = Weight(Previous(j)) + ΔW
 Else
  Weight(j) = [Weight(j)+(Weight(Prev(j)) + ΔW)]/2
  Frequency(i) = Frequency(i) + 1
 End if
End for
```

Therefore, to each category, is attributed, for one given session, its position weight and its frequency. The sessions are thus coded in a table whose lines represent the users (visitors) and with two columns for each category (so that the total number of columns is twice the number of categories): one for the "position weight" and the other for the frequency.

6 Vector Quantization for the behavioral visualization

The representation of the sessions set of the log file on the map is not easily understood.

The vector quantization (QV) has been introduced to create statically represented and economically stored references. The basic idea of this technique results from the fact that in the session based representation space, the coding vectors occupy only a sub spaces [6].
This quantization consists in partitioning the coded sessions in a set of classes and extracting a prototype from each of these classes.
The partitionning is done by Lind Buzo Gray (LBG) algorithm [7].
Finally, to extract a prototype "path" from a class in an uncoded (by « position weight ») form, i.e. in a neural sequence form, the following method is then used:
In the prototypes obtained from the last iteration of LBG algorithm, we only keep neurons whose appearance frequency is higher than a empirically chosen threshold.
The previously selected categories are sorted by ascending order of their position weight.
This is done for all non empty classes and we thus get a number of paths smaller than or equal to the number of classes found by the LBG algorithm (figure 2) .

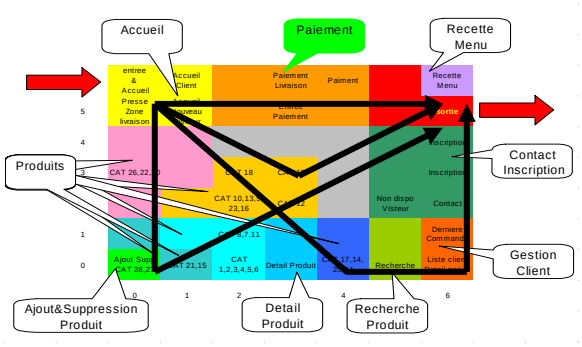


Fig. 2 Paths representation on the map.

7 Recognition

To recognize a given session in comparison with previously found prototypes, we calculate the distorsion which is simply the euclidian distance between the vectors resulting from coding (by position method) of both the session and a prototype.
This computation is done for all prototypes and the one whose distorsion with the session is the smallest is recognized as being the most similar.

8 Results

We used a Log access base of 10.000 sessions describing different behaviors over the visualization of 50 significant page categories. We applied our coding method (position weight) which reorganized the base in a 10.000 × (50 × 2) data table.

With a 16 bytes coding, we got 14 session prototypes.
For the recognition, we used a test base of 1000 sessions labeled by the 14 session prototypes previously found, and obtained an 80% success rate. We think that the mistaken 20% are due to the loss of information induced by the coding.

9 Conclusion

Thanks to a sequence analysis process based on both a new coding method and vector quantization, we could summarize the set of web site visitors navigations into categories representing significant profiles which we have represented on a synthetic map.
This work enabled us to have a first comprehension of the web site visitors whose behaviors are rather difficult to target a priori.

10 References

1. k. Benabdeslem and Y. Bennani, « Classification connexionniste pour l'analyse des comportements d'internautes », *Technical report, LIPN – CNRS,* France, 2000.
2. A. Joshi, "On Mining Web Access Logs". Technical Report, CS Department, University of Maryland, College Park, 1999.
3. k. Benabdeslem, Y. Bennani, E. Janvier « Analyse de données comportementales d'Internautes», *Journée Thématique : Exploration de données issues d'Internet, Institut Galilée, LIPN – CNRS, Villetaneuse*-France, 2 Mars 2001.
4. Benabdeslem K., Bennani Y. et Janvier E. – " Connectionnist approach for Website visitors behaviors mining", Proceedings of ACS/IEEE International Conference on Computer Systemes and Applications, Lebanon 2001.
5. Cadez I., Heckerman D., Meek C., Smyth P., White S. "Visualization of Navigation Patterns on a Web Site Using Model Based Clustering" in proceedings of the KDD 2000.
6. A. Gourinda. "Codage et reconnaissance de la parole par la quantification vectorielle"; thèse de doctorat ; Université Nancy I ; 1988.
7. Linde Y., Buzo A. et Gray R.M., "An algorithm for the VQ design" IEEE, Trans. On Communication, Vol. Com-28, 1980, pp.84-95.
8. T. Kohonen, "Self Organizing Map and Association Memories". *Springer, Vol 8, Springer Verlag*, 1984.

AICCSA'03 : ACS/IEEE International Conference on Computer Systems and Applications

La date : 14 – 18 Juillet 2003

Lieu : University of Tunis

Titre :

Hybrid Connectionist Approach for Knowledge Discovery from Web Navigation Patterns

AICCSA'03: ASC/IEEE International Conference on Computer Sciences and Application, Tunis2003

Hybrid Connectionist Approach for Knowledge Discovery from Web Navigation Patterns

Arnaud Zeboulon
NumSight Consulting SA, Champs Montants 16 C - 2074 Marin - Suisse
arnaud.zeboulon@free.fr

Younès Bennani
LIPN-CNRS, Université Paris 13
99 Avenue J-B. Clément
93430, Villetaneuse, France
younes@lipn.univ-paris13.fr

Khalid Benabdeslem
LIPN-CNRS, Université Paris 13
99 Avenue J-B. Clément
93430, Villetaneuse, France
kbe@lipn.univ-paris13.fr

Abstract

In this article, we apply the "EM" algorithm to learning the parameters of a Markov chain mixture model for clustering navigation sessions on a Web site. Our main contribution is to deduce the model's initial parameters from clusters formed by a hierarchical clustering of a sample of sessions, whose dissimilarity matrix is computed by Dynamic Time Warping. The states of the Markov chains are the neurons of a Kohonen Self Organizing Map, which displays the site as it is seen by the users and also clusters its pages (one neuron corresponding to a cluster of pages). This technique for clustering sessions has been validated on a set of semi-artificial data and the results are excellent. Finally, we tested several criteria for the determination of the optimal number of clusters and concluded that the Akaike Information Criterion was best suited to this problem.

Keywords
Webmining, Clustering, Classification, SOM, Markov Model

INTRODUCTION

A major problem faced by Web site managers is to know what the users are doing on their site, so as to be able to modify it accordingly. For this, the most reliable source of information is the file, provided by the site's server, which records all the data (time, users's IP address, URL of the requested page etc.) regarding the user's requests. This file, called the "Log" file, allows one to reconstitute the navigation "session", which are defined as the sequence of pages seen by the same person during a visit of the site. In this framework, the present work had two goals ([16]): on the one hand, to find a session clustering method, the difficulty being that those sequences are not necessarily of the same length; on the other hand, to find a session recognition technique which, assuming that some navigation «prototypes» are known, allows one to attribute any session to one of these prototypes. From a theoretical point of view, this problem amounts to defining a similarity (or a distance) between two event sequences, the difficulty being again that they usually have different lengths. The problem of recognition, or of definition of a similarity between navigation sessions on a Web site, has to our knowledge never been tackled : the articles that we found about Log file analysis were focusing either on the prediction of future requests (for example [7]) or on adaptive Web sites [12]. Nevertheless, the problem of computing a distance between sequences of different lengths has been treated in other fields, notably in biology [11] and in speech recognition [13] by Dynamic Time Warping (DTW) – or dynamic programming [2]. That is what lead us to try this technique. We found one article [5] dealing with the clustering of Web site navigation sessions. The method proposed in this article – modeling the sessions as a mixture of Markov chains – motivated our work, which adds several original, and we think significant, improvements : a new initialization technique for the algorithm (called "Expectation-Maximization") used to learn the model's parameters; the use of Markov chains with many more states than those used by [5], using a priori clustering and coding of the site's pages by a Self-Organizing-Map (SOM) [9], which thus leads to a much finer clustering of the sessions; and testing several criteria for the determination of the optimal number of clusters, yielding the selection of one of these criteria.

CODING AND VISUALIZATION OF WEB NAVIGATION PATTERNS

We would like to describe a method of exploratory analysis of dynamic behaviors for web site visitors. To validate our work, we use the access log representing thousands of individuals in a commercial web site.
Therefore, our data base includes individuals which represent web site visitors, in which each one is supposed to be interested by a set of pages. Each user (session) will be represented by a succession of pages[5].

Table 1. Format of the navigation file

Session N°	pages
S_1	2 7 4 1 5
S_2	8 4 32
S_3	3 9 23 20

The different numbers in the box "pages" each represent a page of the site.
Actually, the sites are dynamic. That means that the pages are not characterized by a fixed variables as: hierarchical address (URL), the content ....etc. They are represented by a numerical identifiers which have no meaning. For that reason, we have elaborated a novel method of coding sessions from the Log file which consists in characterizing a page by its passage importance, i.e. by its weight of precedence and succession relative to the other pages of the site which appear in the log file.
The principle is to calculate for each page its frequency of precedence and succession over all the other pages and to regroup these frequencies in one matrix : Pages × (previous Pages + next Pages) that we call : quasi-behavioral matrix. We remarked that this coding has a serious problem of size. In fact, if we don't have a enough pages in the site, we can not have a table with enough data to get a robust model. To solve this problem, we proposed to slide the matrix along the log file. In other words, we computed a quasi-behavioral matrix for each time period. This method allowed us to increase the number of samples and to have several examples for each page.

Table 2. Quasi-bihavioral Matrix

Individus (URLs)	jour	E	URL..... URL_N	URL_1..... URL_N	S
E					
URL_1					
URL_2					
URL_i					
URL_N					
S					
E					
URL_1					
URL_2					
URL_i	***k***	***a***	***b***	***c***	***d***
URL_N					
S					
….					
E					
URL_1					
URL_2					
URL_i					
URL_N					
S					

Example:
Let's suppose we have a Log file with 40 URLs over 30 days.
If we slide the matrix along the month day by day, we get a 40 × 30, ie 1200, samples base. We not only characterize the pages by behavioral variables i.e. as the web site visitors perceive the site, but we increase the information about the pages by coding their behavioral characteristics in detail, day by day.
For example, for URL_i , we can code its vector over all the others URLs as follow:
In the k^{th} day of the file, the URL_i appeared *a* times as an entry page, preceded *b* times by the URL_j, succeeded *c* times by URL_l and appeared *d* times as an exit page.
We remember that the log files have an important size and contains relevant information about the web site visitors behaviors.
For the profiles extraction, we use Kohonen's topological map algorithm [9] in order to group the similar pages[9], then we apply an ascendant hierarchical clustering to optimize the classes. Therefore, we talk about clusters of neurons.
The obtained clusters need to be labeled. However, at the time of use of the map, each URL can activate more than one neuron, depending on its passage importance change day after day. Theoretically, if the map is well done, the different neurons which are activated by the same page must be close to one another.
Practically, to label the map in this case, we proceed by a majority vote (Figure1).

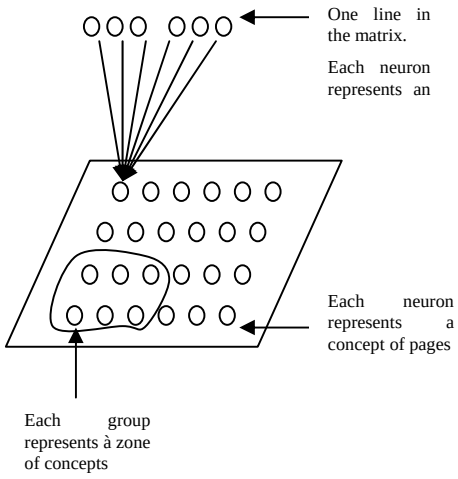


Figure 1. Quasi – bihavioral clustering

CLUSTERING WEB NAVIGATION PATTERNS

The usual distance-based clustering algorithms (hierarchical clustering, k-means, …) require as input data either a fixed-width table of observations-variables, or a similarity (or dissimilarity) matrix for the observations. Here, the number of variables (pages or neurons visited) varies depending on the observations (sessions). Moreover, were we to use a dissimilarity matrix, we would then be faced with the question of which dissimilarity measure (between the observations) to choose. It is obvious that the traditional distances (Euclidean, Hamming etc.) are not suitable because they

require a constant number of variables. Another possibility would be to use a dissimilarity rate computed with DTW. However, this would lead to an unmanageable algorithmic level of complexity for big files. Therefore, we chose another paradigm: clustering based on a stochastic mixture model [10].

A Markov Chain Mixture Model for Clustering

The key idea in model-based clustering is that the observed data are assumed to be generated from a mixture of K "prototype" statistical distributions, each representing a cluster, and within which a certain variability is possible. Statisticians refer to such a model as mixture model with K components. Initially, of course, the model is not known; only the data are available. But it is possible to apply standard statistical techniques to the data to « learn » the model, i.e: the number of components, K; the *a priori* membership probabilities of an observation belonging to a cluster, or the «weights» of the components; the parameters of the individual components. Once the model is learned, one can compute the membership probabilities of an observation for the different clusters and thus infer a partition of the observations. It is also possible to use this approach for recognition of the observations. A detailed mathematical description of a mixture model with K components and the use of the EM algorithm ([6]) to learn its parameters can be found in [8], [10] or [16].

When (as is the case here), the observations are sequences of events (here, of neurons), a natural candidate for the distributions (components) is a Markov chain, whose states correspond to the various neurons on the map's site. That a navigation session on the Web actually satisfies the Markov property is not obvious : indeed, it is not clear that the probability that somebody goes to a page (say j) at click t+1 during a session depends only on the page (say i) on which he was at click t. Nevertheless, without any proof to the contrary, we suppose this assumption is valid to a first approximation. This model thus captures the user's initial request, the dependency between two consecutive requests, and – by virtue of the inclusion of the end state – the last neuron visited.

For the clustering, we propose to model the map (SOM) by a Markov chain (MC) (Figure 2). In other terms:

A neuron in SOM represents a state in MC

A connexion in SOM represents a transition in MC.

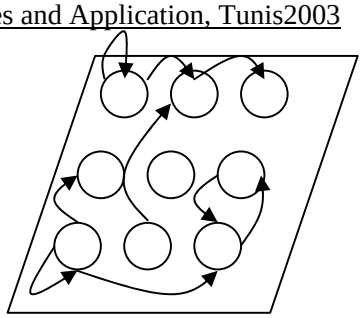


Figure 2. SOM based Markov Model

Importance of Initialization in Mixture Model

It has been proven (see for example [15]) and is well known that the EM algorithm converges only towards a *local* maximum of the likelihood. Consequently, the initialization phase of the algorithm is known to be very important: a poor initialization could lead to a poorly-fitted model and hence to a poor clustering. We propose a new initialization method, which seems to us much more pertinent than the one used in [5]. We have verified that the dissimilarity rate between sequences of different lengths, computed by a DTW algorithm, is a very good measure [16]. We had then the idea to extract a sample of observations randomly drawn from the complete data set, to compute via DTW the dissimilarity matrix of the sample's elements (taking the elements two by two), and to carry out a hierarchical clustering using this matrix. By cutting the resulting dendogram at a level such that K clusters appear, we then compute the parameters associated with these K classes and use those values to initialize the EM algorithm. In practice, the time needed to compute the dissimilarity matrix for of a sample of 500 observations was around 10 minutes, which was quite acceptable, and therefore we chose samples of this size for validating the method.

Determination of the Optimal Number of Clusters

The clustering algorithm presented above works well in general but has one major limitation : it requires specifying the number of clusters K, whereas in practice, this is seldom known. Therefore, we would like to be able to find, also automatically, the optimal (or "natural") number of groups in the data set. The standard statistical solution to this problem is to introduce a (numerical) criterion which measures the fit of the model (and hence the number of components) to the analyzed data [8]. We then compute this criteria for several values of K and the one that maximizes it is taken as the optimal number of clusters (and the associated partition considered to be the best). Several criteria have been proposed in the literature. The first was the Akaike Information Criterion [1] or "AIC", the Bayesian Information Criterion [14] or "BIC", and finally, Cadez et al. [5] used a score reflecting the average number of bits required to encode a category of pages (neuron in our case) requested by the user.

APPLICATION TO CLUSTERING OF WEB NAVIGATION SESSIONS

Data Used for Validation

Regarding data, a lack of labeled data has hindered our efforts to measure the performance of the algorithm. Indeed, we only had "raw" Log files, in which navigation sessions were neither assigned to a particular "prototype" nor grouped into clusters. To overcome this problem, we developed a semi-artificial session generator. We started out with real data, drawn from a Log file of the site www.123Credit.com over a period of one month (30 days). This yielded 37174 sessions, with a total of 40 different pages (URLs) visited during these sessions.

Extraction of Sessions Prototypes

From these 37174 sessions, 10 prototypes (representatives) were extracted by the technique previously used in our system [17] and described in [4]. Starting with these 10 prototypes, the generator created 1000 sessions per prototype (so 10000 sessions total), by randomly perturbing the protoypes with deletions, insertions and replacements of neurons. Besides, a sample of 500 sessions has been randomly drawn from the 10000 (for the initialization of the EM algorithm). The 10000 sessions generated in this manner were «naturally» labeled (the first 1000 by the 1st prototype, the next 1000 by the second prototype etc.), which allowed us to measure the quality of the clustering (Figure3), and to test criteria for the determination of the optimal number of clusters. The results of those tests are described in the paragraphs below.

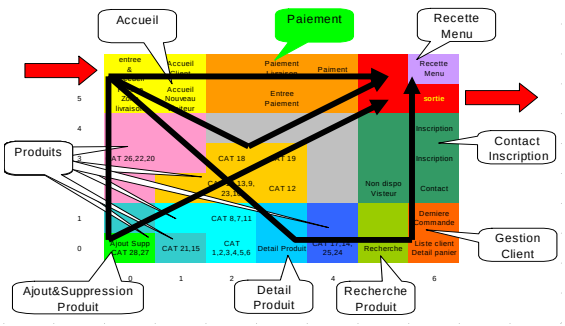


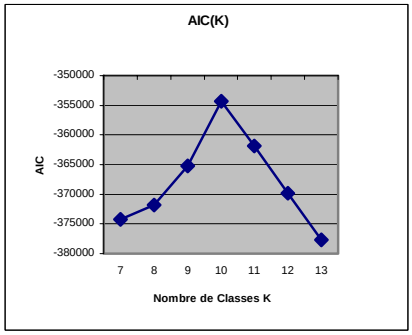
Figure 3. Mapping of a commercial site with Representation of user's trajectory

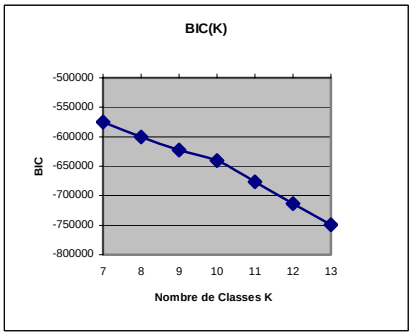
Sessions Clustering and Recognition

The average success rate for the clustering was measured by counting the number of sessions generated from a given prototype and actually assigned by the program to the corresponding cluster. With this definition, we obtained an average of 99.23% ([99.04%, 99.38%]) of well classified sessions. The values between brackets is the confidence interval for a 95% confidence level, computed as described in [3]. Besides, we also measured the success in the presence of cycles (i.e. the same neuron appearing twice in a session) in the reference prototypes and found a comparable value (99.3 %) ([99.12%, 99.45%]). Finally, in both cases - prototypes with and without cycles -, the marginal probabilities of membership in the clusters (ie the weights of the model's components) were all approximately equal to 0.1 within a few percent. We attribute the quality of those results on the one hand to our initialization technique for the EM algorithm, which makes it start from «good» values of the parameters, and on the other hand to the relatively high number (more than 50) of possible states (or neurons) in the Markov chains, which leads to "high resolution" clustering.

Determination of the Number of Clusters

The two most used criteria being the AIC and the BIC, we focused our trials on them, and on the criterion proposed by [5] (that we shall note "CIC", for "Cadez Information Criterion"). The data being partitioned in 10 clusters, we compared the values given by them with K = 10 (tests were carried out for values of K ranging from 7 to 13). The AIC was the only one of the three with this property, and it is perhaps best suited for this type of clustering. In contrast, the BIC was decreasing for all K and thus seemed inappropriate. As for the CIC, although it doesn't reach a maximum, its slope did change abruptly at K=10 and it could thus perhaps be used in this fashion. Nevertheless, this would be more complicated and have greater risk of error than the simple observation of a maximum allowed by the AIC. Thus we conclude that the AIC is the best criterion for our problem (Figure4).





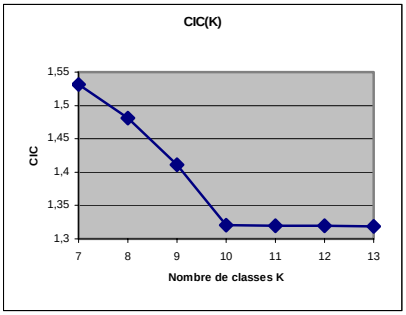


Figure 4. Information Criterions for the best number of clusters

CONCLUSION

The clustering technique presented here offers several advantages. First, the use of many neurons as states of the Markov chains (as opposed to a small number of categories of pages in the case of [5]) as well as the initialization method that we introduced yielded highly accurate clustering. Second, the statistical tools used are relatively easy to implement. Finally, the algorithm is both fast and scalable to very large data sets : in our tests, clustering 10000 sessions only took a few seconds. This work could be continued in at least two directions: extracting representatives from the obtained clusters and on-line prediction of the end of a navigation session based on its beginning (for example by analyzing the transition matrix of the cluster to which the session most likely belongs).

REFERENCES

[1] Akaike H.: A new look at the statistical identification model. IEEE Transactions on Automatic Control, 19, (1974) 716-723.

[2] Bellman R.: Dynamic Programming. Princeton University Press, Princeton, (1957).

[3] Bennani Y.: Multi-expert and hybrid connectionnist approach for pattern recognition : speaker identification task . International Journal of Neural Systems, Vol. 5, No. 3, (1994) 207-216.

[4] Benabdeslem K., Bennani Y., Janvier E.: Connectionnist approach for Website visitors behaviors minin. Proceedings of ACS/IEEE International Conference on Computer Systemes and Applications, Lebanon (2001).

[5] Cadez I., Heckerman D., Meek C., Smyth P., White S.: Visualization of Navigation Patterns on a Web Site Using Model Based Clustering. In proceedings of the KDD (2000).

[6] Dempster A.P., Laird N.M., Rubin D.B.: Maximum likelihood for incomplete data via the EM algorithm. Journal of the Royal Statistical Society, B, 39, (1977) 1-38.

[7] Deshpande M., Karypis G.: Selective Markov Models for predicting Web-page accesses. 1st SIAM conference on Data Mining, Chicago, Illinois, (2001).

[8] Fraley, C. and Raftery, A.: How many clusters ? Which clustering method ? Answers via model-based cluster anlysis. Computer Journal, 41 (1998) 578-588.

[9] Kohonen T.: Self-Organizing Map. Springer, (1995).

[10] McLachlan G.and Basford K.: Mixture Models : Inference and Applications to Clustering. Marcel Dekker, (1988).

[11] Needleman, S.B. and Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Mol. Biol., 48 (1970) 443-453.

[12] Perkowitz M., Etzioni O.: Towards adaptative web sites : conceptual framework and case study. Artificial Intelligence Journal, 118 (2000) 1-2.

[13] Sakoe H.: Two-level DP matching - a dynamic programming based pattern matching algorithm for connected word recognition. IEEE Transactions on Acoustic Speech and Signal Processing, 27(6) (1979) 588-595.

[14] Schwarz G.: Estimating the dimension of a model. Annals of Statistics, 6, (1978) 461-464.

[15] Wu C.F.J.: Annals of Statistics, 11(1), (1983) 95-103.

[16] Zeboulon A.: Reconnaissance et classification de séquences. DEA127's internship report, Université Paris 9, (2001).

[17] Benabdeslem K., Bennani Y., Janvier E, "Visualization and analysis of web navigation data" LNCS2415 (Springer), pp 486-491, Madrid, 2002.

8 Bibliographie

[01] F. Paradis – Information Extraction and Gathering for Search Engines: The Taylor Approach. *In RIAO2000, Conference on "Content-Based Multimedia Information Access", pp78-85,Australia*, 2000.

[02] L. Chittaro and R. Ranon – Adding Adaptive Features to Virtual Reality Interfaces For E-Commerce. *In Proceeding International Conference On Adaptive Hypermedia and Adaptive Web Based Systems (AH2000), Lecture Notes in Computer Science, Springer, Berlin*, 2000.

[03] M. Lebbah, S. Thiria and F. Badran - Topological Map for Binary Data , *ESANN 2000 , Bruges, pp26-27-28, Proceedings*, April, 2000

[04] K. Abe – Concept – Based Search and User Modeling in Information Retrieval Based on Human – Machine Dialogue. *In RIAO2000, Conference on "Content-Based Multimedia Information Access", Japan*, 2000.

[05] B. Trousse - Evaluation of the Prediction Capability of a User behaviour Mining Approach for Adapative Web Sites. *In RIAO 2000, 6th Conference on "Content-Based Multimedia Information Access", College de France, Paris, France, April pp12-14*, 2000.

[06] E. Selberg and O. Etzioni – On the Instability of Web Search Engines. *In RIAO2000, Conference on "Content-Based Multimedia Information Access", USA*, 2000.

[07] A. Joshi – On Mining Web Access Logs. *Technical Report, CS Department, University of Maryland, College Park*, 1999.

[08] T. Jörding - A Temporary User Modeling Approach for Adaptive Shopping on the Web. *accepted for the " 2nd Workshop on Adaptive Systems and User Modeling on the WWW" at UserModeling'99*, Banff, Canada, 1999.

[09] L. Ardissono and A.Goy – Tailoring the interaction with Users in Electronic Shops. *In Proceeding of the 7th International Conference on User Modeling (UM97)*, Banff, MA, 1999.

[10] J. Mena – Data Mining your Website. *Digital Press, Butterworth – Heinemann*, 1999

[11] S. Kaski – Fast winner search for SOM – Based Monitoring and retrieval of high – Dimensional Data. *In Proceeding of ICANN'99, 9th International Conference on Artificial Neural Networks, Edinburg, UK, pp7 – 10*, 1999.

[12] L. Ardissono – Adaptive Web stores. *In Proceeding of the Agents'99 Workshop: Agent-Based Decision-Support for managing the Internet-Enabled Supply-Chain, pp.99-13*,Seattle, WA, May 1999.

[13] L. Ardissono and A. Goy - Tailoring the Interaction With Users in Electronic Shops. Gzipped PostScript. *In Proceeding of 7th Conference on User Modeling, pp. 35-44, Springer-Verlag* (http://www.springer.de/), *Banff, Canada*, 1999.

[14] S. Kaski – Dimensionality Reduction by Random Mapping: Fast similarity Computation for Clustering. *In Proceeding of IJCNN'98, 1998 IEE International joint conference on neural networks, Anchorage, Alaska*, 1998.

[15] M. Jaczynski and B. Trousse - WWW Assisted Browsing by Reusing Past Navigations of a Group of Users. *In Advanced in Case-based Reaosning, 4th European Workshop on Case-Based Reasoning , Lecture Notes in Artificial Intelligence, 1488, pp160-171*, 1998.

[16] H. Mannila - Methods and Problems in Data Mining (A tutorial). *In Proceedings of International Conference on Database Theory (ICDT'97), Delphi, Greece, , F. Afrati and P. Kolaitis (ed.), pp 41-55*, January 1997.

[17] M. Klemettinen, H. Mannila, and H. Toivonen - A Data-Mining Methodology and its Application to Semi-Automatic Knowledge Acquisition. *In Proceedings of the 8th International Conference and Workshop on Database and Expert Systems Applications (DEXA'97), pp670 - 677, Toulouse, France,* September 1997.

[18] T. Kindo and al - Adaptive Personal Information Filtering System that Organizes Personal Profiles Automatically. *IJCAI 97, pp716-721*, 1997.

[19] T. Kohonen – Exploration of Very Large Databases by Self – Organizing Maps. *In proceeding of ICNN'97, International Conference on Neural Networks*, 1997.

[20] B. Raskutti, A. Beitz and B. Ward – A feature Approach to Recommending Selections Based on Past Preferences. *User modeling and user adapted interaction 7: pp179 – 218*, 1997.
[21] N. Ye – Neural Approach to user modeling and intelligent interface: A review and reappraisal. *International journal of human computer interaction, 9(1), pp3 – 23*, 1997.
[22] Q. Chen and A.F. Norcio – Modeling a User's Domain Knowledge with Neural Networks. *International journal of human computer interaction, 9(1), pp25 – 40*, 1997.
[23] M. Dimitrova and al – Neural Networks for Classification and Recognition of Individual Users in Adaptive Human Computer Interaction. *In Proceeding of the 12th IEEE International Symposium on intelligent control, pp101 – 106*, 1997.
[24] I. Bhandari and al – Advanced Scout: Data Mining and Knowledge Discovery in NBA Data. *Data Mining and Knowledge Discovery, 1, pp121 – 125*, 1997.
[25] M. Pazzani and D. Billsus – Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine learning 27, pp313 – 331*, 1997.
[26] T. Joachims and al – WebWatcher:A tour Guide for the Wold Wide Web. *In Proceeding of the 5th International Joint Conference On Artificial Intelligence (IJCAI'97). Edited By Martha E. Pollack, distributed by Morgan Kaufmann Publishers Inc, San Francisco, CA, pp770 – 775*, 1997.
[27] B. Mobasher - Web Mining: Pattern Discovery form World Wide Web Transactions A Research Overview. *Présentation par B. Mobasher du projet Webminer du laboratoire de recherche de l'université du Minnesota,* juillet1997 http://www-users.cs.umn.edu/~mobasher/webminer.html
[28] B. Raskutti and al – A feature – Based Approach to Recommending Selections Based on Past Preferences. *User Modeling and User – Adapted interaction, 7, pp179 – 218*, 1997.
[29] F. Anouar – Modélisation Probabiliste des Cartes Auto – Organisées : Application en Classification et en Régression. *Thèse de doctorat d'Informatique, CNAM*, 1996.
[30] Y. Lechevallier and al – Statistisques et Méthodes Neuronales, *Dunod*, 1996.
[31] D. Akoumianakis and C. Stephanidis – User Modeling for Adaptive Interface Design. *Symbiosis of Human artifact, pp1071- 1076*, 1995.
[32] R. Armstrong and al – WebWatcher: A Learning Apprentice for the World Wide Web. *AAAI Spring Symposium on information gathering from Heterogeneous, Distributed Environments*, 1995.
[33] J.F. Jodouin – Les Réseaux de Neurones : Principes et Définitions. *Hermes, Paris*, 1994.
[34] J.F. Jodouin – Les Réseaux Neuromimétiques: Modèles et Applications. *Hermes, Paris*, 1994.
[35] A. Jenning and H. Higuchi – A User Model Neural Network for a Personal News Service. *User modeling and user adapted interaction 3: pp1 – 25*, 1993.
[36] M. F.McTear – User Modeling for Adaptive Computer Systems: A survey of Recent Developments. *Artificial intelligence review 7, pp157 – 184*, 1993.
[37] J. Finlay and R. Beale – Pattern Recognition and Classification in Dynamic and Static User Modeling. *In Neural Networks and Pattern Recognition Human – Computer Interaction, R. Beale et J. Finlay éditeurs, Elis Horwood*, 1992.
[38] R. Eller and S. Garberry – A Meta – Rule Approach to Flexible plan Recognition in Dialogue. *User Modeling and User – Adapted Interaction 1(4)*, 1991.
[39] A. F. Norcio and J. Stanley – Adaptive Human – Computer Interfaces: A Literature Survey and Perspective. *IEEE transactions on systems, man, and cybernetics, Vol 19, No2*, 1989.
[40] D.N. Chin – KNOME: Modeling What the User Knows. *User Models in Dialog Systems, Spring Verlag*, London, 1989.
[41] R. Kass – Modeling the User in Natural Language Systems. *Computational Linguistics (Special issue on user modeling) 14(3), pp5 – 22*, 1988.
[42] H.Ritter and K. Schulten – Convergence properties of Kohonen's topology conserving maps: fluctuations, stability and dimension selection. *Biological Cybernetics, 60, pp59 – 71*, 1988.
[43] S. Garberry – Modeling the User's plans and Goals. *Computational Linguistics (Special issue on User Modeling), 14(3), pp23 – 37*, 1988
[44] D. Benyon and al – System Adaptivity and the Modeling of Stereotypes. *In Bullinger, H.J. and Shackel, B.(eds), Human – Computer Interaction: INTER – ACT'87, North Holland, Amsterdam*, 1987.

[45] W. Hoeppner and al – Talking it Over: The Natural Language Dialogue System HAM – ANS. *In Bloc, L. and Jarke, M.(Eds), cooperative Interfaces to Information Systems, Springer, New York*, 1986.
[46] T. Kohonen – Self Organizing Map and Association Memories. *Springer, Vol 8, Springer Verlag*, 1984.
[47] E. Rich – Users are Individuals: Individualizing Users Models. *International Journal of Human – Machine Studies, 18, pp199 – 214*, 1983.
[48] F. Rosenblatt – The perceptron: a Perceiving and Recognizing Automaton. *Project PARA, Cornell Aeronautical Lab. Report pp85 – 460-1*, 1959.
[49] A. Ribert & al – An Incremental Hierarchical Clustering. Vision Interface'99, Trois rivières, Canada, 1999.
[50] J.Blackmore – Visualizing High-dimensional Structure with the incremental Grid Growing Neural Network. Technical report, Departement of computer sciences of the university of Texas, Austin, 1995.
[51] T.Martinez & K.Schulten – A Neural Gas Network Learns Topologies. Artificial Neural Networks, Elsevier Science Publishers B.V, pp 397 – 402, 1991.
[52] B.Fritz – Growing Cell Structures-A self-Organizing Network for Unsupervised Learning. Neural Networks, Vol 7, N°9, pp 1441-1460, 1994.
[53] Y.Linde, A.Buzo, RM.Gray – An Algorithm for the Vector Quantization design. IEEE,Trans. On Communication, Vol. Com-28, pp 84-95, 1980.
[54] R.Bellman – Dynamic Programming. Princeton University Press, Prinston, 1957.
[55] CR.Anderson & al – Adaptive Web navigation for wireless devices. 17th International joint Conference on Artificial Intelligence (IJCAI-01), 2001.
[56] M.Deshpande & al – Selective Markov Models for predicting Web-pages accesses. 1st SIAM (Society for Industrial and Applied Mathematics) conference on Data Mining, Chicago, Illinois, April7, 2001.
[57] CFJ. Wu – Annals of statistics. 11(1), pp 95-103, 1983.
[58] I. Cadez & al – Visualization of Navigation Patterns on a Web Site Using Model Based Clusterig. In proceeding of the KDD'00, 2000.
[59] AP. Dempster, NM Laird, DB. Rubin – Maximum liklihood for incomplete data via the EM algorithm. Journal of the statistical Society, B, 39, pp 1 – 38, 1977.
[60] Y. Bennani – Multi – expert and hybrid connectionnist approach for pattern recognition : speaker identification task. International journal of neural systems, Vol 5, N° 3, pp 207 – 216, September 1994.
[61] H. Akaike – A new look at the statistical identification model – IEEE transactions on automatic control, 19, pp 716 – 723, 1974.