Reconnaissance de caractères par méthodes markoviennes et réseaux bayésiens

Khalid Hallouli

Thèse

présentée pour obtenir le grade de docteur de l'Ecole Nationale Supérieure des Télécommunications
Spécialité : **Signal et Images**

Khalid HALLOULI

Reconnaissance de caractères par méthodes markoviennes et réseaux bayésiens

Soutenue le 05 mai 2004 devant le jury composé de :

Bruno TACONET	President
Nicole VINCENT	Rapporteur
Hubert EMPTOZ	Rapporteur
Edouard GEOFFROIS	Examinateur
Laurence LIKFORMAN-SULEM	Directeur de thèse
Marc SIGELLE	Directeur de thèse

À toute ma famille
à ma femme ...

Remerciements

Je tiens à exprimer toute ma reconnaissance à ceux qui ont contribué à ce travail, en particulier :

Mes directeurs de thèse Madame *Laurence Likforman-Sulem* et Monsieur *Marc Sigelle*, de m'avoir bien encadré et dirigé attentivement mes travaux de recherche, pour leurs disponibilités, leurs efforts pour la réalisation de ce travail.

Monsieur *Bruno Taconet* professeur à l'Université du Havre, qui m'a fait l'honneur de présider ce jury de thèse.

Madame *Nicole Vincent* professeur à l'université de Paris 5, pour avoir accepté d'être rapporteur de cette thèse, pour ses conseils instructifs qu'elle m'a donné et pour ses remarques contribuant ainsi à l'amélioration de ce manuscrit.

Monsieur *Hubert Emptoz* professeur à l'INSA de Lyon, pour avoir accepté d'être rapporteur de cette thèse et pour ses précieux conseils

Monsieur *Edouard Geoffrois* pour avoir accepté de faire partie de ce jury et pour ses vifs conseils qui m'ont été fort utiles

Je tiens également à remercier ici *Gérard Chollet*, *Maurice Charbit* pour leurs conseils durant ce travail de thèse, mes collègues et amis du dépatement TSI, plus particulièrement Eduardo, Thomas, Raphaël, Belgacem.

Une fois de plus je remercie mon papa, ma grande mère, Saadia, ma maman, ma femme Sihame et Rachida pour leurs soutiens et leurs encouragements durant ces années d'études.

Également je tiens à remercier mes amis : Jourani, Wahid, Rochdi, Hani, ...

Résumé

Cette thèse porte sur la reconnaissance de caractères imprimés et manuscrits par méthodes markoviennes et réseaux bayésiens.

La première partie consiste à effectuer une modélisation stochastique markovienne en utilisant les HMMs classiques dans deux cas : semi-continu et discret. Un premier modèle HMM est obtenu à partir d'observations de type colonnes de pixels (HMM-vertical), le second à partir d'observations de type lignes (HMM-horizontal). Ensuite nous proposons deux types de modèles de fusion : modèle de fusion de scores qui consiste à combiner les deux vraisemblances résultantes des deux HMMs, et modèle de fusion de données qui regroupe simultanément les deux observations lignes et colonnes. Les résultats montrent l'importance du cas semi-continu et la performance des modèles de fusion.

Dans la deuxième partie nous développons les réseaux bayésiens statiques et dynamiques, l'algorithme de Jensen Lauritzen Olesen (JLO) servant comme moteur d'inférence exacte, ainsi que l'apprentissage des paramètres avec des données complètes et incomplètes. Nous proposons une approche pour la reconnaissance de caractères (imprimés et manuscrits) en employant le formalisme des réseaux bayésiens dynamiques. Nous construisons certains types de modèles : HMM sous forme de réseau bayésien dynamique, modèle de trajectoire et modèles de couplages. Les résultats obtenus mettent en évidence la bonne performance des modèles couplés.

En général nos applications nous permettent de conclure que l'utilisation des réseaux bayésiens est efficace et très prometteuse par le fait de modéliser les dépendances entre différentes observations dans les images de caractères.

Mots-clés : Modèles de Markov cachés, Algorithme EM, Intelligence artificielle, Réseau Bayésien Statique et Dynamique, Reconnaissance de caractères imprimés et manuscrits, Inférence.

Abstract

This thesis concerns printed and handwritten character recognition using Markovian models and Bayesian Networks.

The first part consists of combining two hidden Markov models using semi-continuous and discrete HMMs. The first HMM is constructed using the vertical flow of writing (vertical HMM), while the second is obtained using the horizontal flow (horizontal HMM). Then we propose two fusion schemes : a decision fusion model which combines likelihoods resulting from both vertical and horizontal HMMs, and a fusion data model which consists in a unique HMM which takes as observations both the horizontal and vertical flows of writing. The results show the importance of the semi-continuous case and the performance of fusion models.

In the second part, we introduce static and dynamic bayesian networks. Then we present the Jensen Lauritzen Olesen algorithm (JLO) used for exact inference, and parameters learning with full and partial observability. The new approach we propose for character recognition use the formalism of dynamic bayesian networks (DBN). We build and test various models : single-flow HMMs, auto-regressive HMMs by coupling dynamically observations and coupled models coupling the two flows of writing. The results obtained show that coupled models perform better (up to 6%) than single-flow models.

Table des matières

Table des figures

Liste des tableaux

Introduction

Les techniques liées au traitement de l'information connaissent actuellement un développement très actif en liaison avec l'informatique et présentent un potentiel de plus en plus important dans le domaine de l'interface homme-machine. L'écrit restera l'un des grands fondements des civilisations et le mode par excellence de conservation et de transmission du savoir. La reconnaissance de l'écriture (RE) est un domaine vaste qui constitue un sous-ensemble des systèmes de la reconnaissance des formes (RF). Ces systèmes sont la première étape d'un processus de compréhension de notre univers dans le cadre global de la communication homme-machine. La reconnaissance de l'écriture manuscrite ou imprimée reste encore un sujet de recherche et d'expérimentation, le problème n'est pas encore entièrement résolu bien que l'on sache atteindre des taux assez élevés dans certaines applications pour lesquelles soit le vocabulaire est limité, soit la fonte est unique ou en nombre restreint.

Il existe cependant plusieurs domaines pour lesquels la reconnaissance de l'écriture est appliquée avec un certain succès : la bureautique avec les systèmes OCR (Optical Character Recognition), le tri automatique du courrier, le traitement automatique de dossiers administratifs, des formulaires d'enquêtes, ou encore l'enregistrement des chèques bancaires. La reconnaissance de l'écriture manuscrite est beaucoup plus complexe que celle de l'écriture imprimée due à son extrême variabilité : variabilité des formes, des espacements entre mots et caractères, fluctuation des lignes. Cependant, même pour les documents imprimés, les dégradations affectent très rapidement les performances de reconnaissance. C'est le cas pour les photocopies, les télécopies et les documents typographiés au travers d'un papier carbone etc.... D'autre part, il existe tout un domaine d'application pour lequel la reconnaissance automatique des caractères, même pour les documents imprimés, n'est pas du tout résolue : il s'agit des documents patrimoniaux provenant des bibliothèques et des archives. La reconnaissance des caractères y est nécessaire pour l'accès au contenu et l'indexation automatique. Cet accès au contenu, et non pas seulement à l'image, permet de valoriser les collections de documents imprimés anciens (XVIème

siècle) ou plus modernes (registres) en les mettant à la disposition du public ou des chercheurs. Or ces documents ont subi diverses dégradations dues au temps et au procédés d'impression, d'autre part certaines formes de caractères sont caractéristiques d'une époque : des systèmes de reconnaissance spécifiques doivent être généralement construits robustes au bruit et adaptés aux formes considérées. Des recherches sont en cours pour traiter l'écriture manuscrite de ces documents.

Une grande partie des problèmes de reconnaissance de l'écrit est traitée par des méthodes statistiques du fait que ces méthodes ont une certaine capacité d'intégration du contexte, de la variabilité des observations et d'absorption du bruit. Parmi les approches statistiques, les approches stochastiques et parmi elles les approches markoviennes sont largement utilisées pour la reconnaissance automatique de la parole (RAP). Cependant, le signal de parole est mono-dimensionnel alors que les images sont bi-dimensionnelles. Aussi, les chercheurs sur l'écrit se sont inspirés des idées leurs confrères de la parole en adaptant des techniques telles que les modèles de Markov cachés aux problèmes spécifiques de la reconnaissance de l'écriture. Ceci suppose la recherche d'un ensemble de séquences d'observation adéquat (nous n'en ferons pas exception dans la première partie de ce rapport).

Une approche courante du problème de la reconnaissance de l'écriture consiste à utiliser un processus stochastique mono-dimensionnel de façon similaire au traitement de la parole, et à développer des outils qui donnent la probabilité d'appartenance d'une forme (caractère) à une classe (modèle) en fonction de la distorsion subie par cette forme. Les HMMs offrent une façon commode pour réaliser ce point. Le processus de RAP traite un signal temporel, et le processus RE traite lui, avec la même approche fondée sur l'hypothèse markovienne, un signal dépendant de la coordonnée horizontale. Ce passage de l'image (bidimensionnelle) à une séquence 1D d'observations est souvent accompagné d'une perte d'information contextuelle qui nuit à la qualité de la modélisation. L'application classique des HMMs consiste à effectuer l'apprentissage des modèles. Ce dernier consiste à répartir le mieux possible les observations dans les états. Donc un état donné se voit attribuer un ensemble de sous-séquences dont il se sert pour estimer la probabilité d'observation. De plus le chemin le plus probable ou l'estimation de la partie cachée est calculé par programmation dynamique en faisant des maximisations locales à chaque instant t. Or le calcul de ce chemin revient à emprunter des sous-séquences issues via l'apprentissage. Nous explorons aussi, dans le deuxième chapitre, le cadre des HMMs par des approches de type fusion pour la reconnaissance des caractères imprimés dégradés et manuscrits : fusion de scores de classifieurs HMMs classiques construits indépendamment, fusion de données en couplant les observations issues de séquences

différentes. Ces premières approches de type couplage améliorent les résultats de reconnaissance, ce qui encourage à developper ce point.

La nature 2D de l'image des caractères permet de penser que des améliorations peuvent être apportées aux systèmes de reconnaissance. Dans ce sens des travaux sont apparus parmi lesquels les PHMMs (pseudo- ou planar-HMMs [Saon(1997)], les couplages de HMMs ([Brand(1997)]), les champs de Markov et récemment les réseaux bayésiens (**BN**) qui sont une généralisation des modèles HMMs. Une extension des réseaux bayésiens permettant de traiter des séquences d'observations, sont les réseaux bayésiens dynamiques. Ceux ci ont tout d'abord été introduits en reconnaissance automatique de la parole avec l'objectif de modéliser plus finement le signal et/ou de le décomposer en plusieurs flux d'observations. Dans ce dernier cas, l'hypothèse de départ est que chaque flux étant extrait dans des sous-bandes fréquentielles différentes, le bruit n'affecte pas les sous bandes avec la même intensité.

Ainsi un réseau bayésien (réseau probabiliste, ou Bayesian Belief network) est un modèle représentant des connaissances incertaines sur un phénomène complexe, permettant à partir des données un véritable raisonnement. Ainsi un BN est un graphe acyclique orienté dont les nœuds sont des variables aléatoires qui peuvent avoir un certain nombre d'états discret ou continu selon une loi continue. La structure graphique du réseau représente les relations entre variables (plus précisément les independances conditionnelles). Les réseaux bayésiens statiques sont largement appliqués depuis les années 1996 dans le domaine du diagnostic médical, le domaine de la sécurité (détection des fraudes, fiabilité), l'ingénierie des connaissances.

En ce qui concerne la reconnaissance des images de caractères, les réseaux bayésiens dynamiques vont nous permettre de construire des modèles bi-dimensionnels dont la structure graphique va reflèter les relations entre variables d'états et/ou variables d'observations. Les observations vont être issues de deux flux distincts : l'un obtenu par balayage horizontal de l'image du caractère, l'autre obtenu par un balayage vertical. Cette approche fournit ainsi un nouveau cadre théorique pour la reconnaissance des caractères en s'affranchissant des hypothèses liées aux modèles HMMs et qui concernent surtout l'indépendance des observations entre elles. Les modèles de couplage à l'aide des réseaux bayésiens dynamiques que nous proposons dans la deuxième partie de ce rapport, prennent en compte la corrélation entre les lignes et les colonnes. Ainsi, on peut coupler aux mêmes instants ou à des instants voisins :

– les états cachés associés aux deux chaînes

– les observations, dynamiquement dans le temps (modèles de trajectoires)

– les états et les observations

Ceci laisse de grandes possibilités de choix pour obtenir et tester des structures variées de réseau. L'amélioration due au couplage est sensible, en particulier pour les modèles de trajectoire couplés où les observations sont liées dans le temps.

Plan de la lecture de ce rapport

Nous commençons ce rapport par une introduction générale. Nous exposons la problématique de la reconnaissance des caractères imprimés. Nous donnons quelques définitions et descriptions des termes de l'**OCR**. Nous présentons quelques techniques existantes d'analyse et de reconnaissance en rapport avec nos objectifs.
Dans le second chapitre, nous expliquons la décision bayésienne, les critères d'estimation pendant l'apprentissage, l'algorithme EM, les modèles de Markov cachés (HMMs) ainsi que l'apprentissage et la reconnaissance. Ensuite nous donnons des évaluations numériques.
Dans le troisième chapitre nous exposons les réseaux bayésiens (BN), ainsi que l'inférence dans les BNs (technique d'élimination des variables, construction de l'arbre de jonction, inférence sans et avec un ensemble d'observation (évidence)) et l'apprentissage dans les BNs.
Les réseaux bayésiens dynamiques (DBNs) et les résultats numériques font l'objet du quatrième chapitre. Dans ce dernier, nous donnons une représentation d'un réseau bayésien dynamique, la représentation des HMMs comme des DBNs, l'inférence dans les DBNs et enfin les applications de ce type de modèles à la reconnaissance des chiffres manuscrits et des caractères imprimés dégradés.
Enfin, nous donnons une conclusion générale sur ce travail ainsi que des perspectives possibles pour poursuivre cette recherche.

Chapitre 1

Techniques classiques de reconnaissance

1.1 Introduction

Chaque information écrite peut être reprise dans une chaîne de traitement informatisée avec différents objectifs (gestion électronique de documents, rédaction, édition de rapports, archivage, tri $\cdots$). La reconnaissance optique de caractères (**OCR**) est une opération permettant d'effectuer la transformation d'un texte sous forme image en un texte sous forme d'un fichier informatique de façon rapide et automatique. Les systèmes de reconnaissance de caractères sont différents suivant que l'on traite les caractères isolément ou non. Dans le premier cas, il s'agit de la reconnaissance de la forme des caractères sans prise en compte du contexte. Dans le second, la reconnaissance prend en considération le contexte lexical ou celui de la structure du texte.

1.2 Notion d'OCR

La reconnaissance optique de caractères est un procédé permettant de récupérer les symboles dans les images de textes numérisées. Les images (par exemple d'une télécopie) consistent en des matrices de pixels. La tâche d'un OCR consiste à segmenter les images en lignes, mots, caractères puis à effectuer la reconnaissance des symboles. Il en résulte une transcription textuelle de l'image par laquelle des traitements automatiques sont possibles : recherche de mots, de noms, résumé, $\cdots$ etc. En général, l'OCR concerne le traitement d'un document numérisé. L'OCR connaît des applications pratiques dans plusieurs domaines :

- les banques pour l'authentification de chèques, et les assurances pour la vérification de clauses de contrats.
- la Poste pour la lecture des adresses et le tri automatique du courrier.
- les télécommunications pour l'échange à distance de fichiers informatisés.

Jusqu'à maintenant, il n'existe pas de système universel d'OCR mais des voies d'approches qui dépendent du type des données traitées et de l'application envisagée. Pour la reconnaissance on a toujours le problème concernant le style calligraphique (fonte) : monofonte, multifonte ou omnifonte. Un système est monofonte s'il ne connaît que l'alphabet d'une seule fonte. La reconnaissance est simple puisque l'alphabet représenté est réduit. Un système multifonte est capable de reconnaître plusieurs fontes parmi un ensemble de fontes préalablement apprises. Dans ce cas la reconnaissance est difficile puisque le système doit identifier les différentes formes, et gérer plus d'ambiguïtés. Un système est dit omnifonte s'il est capable de reconnaître toutes les fontes sans les avoir apprises, ce qui trop difficile à réaliser puisqu'il y a au moins 3000 fontes qui ne sont pas lisibles parfois par des utilisateurs[1].

1.3 Analyse

Le but de l'analyse est d'extraire les propriétés caractéristiques de l'image du caractère et de l'exprimer sous forme numérique ou symbolique. Ces caractéristiques doivent être invariantes aux déformations possibles intra-classe. Certains prétraitements peuvent être nécessaires. Parmi ces prétraitements, on trouve :

- le *redressement* des caractères. L'idée est de rendre verticaux les longs traits quasi-verticaux à l'aide d'une transformation qui modifie l'abscisse x des pixels d'écriture en fonction de l'angle d'inclinaison. Celui-ci est obtenu après élimination des traits horizontaux [Bozinovic et Srihari(1989)].
- la *rotation* permet de tourner les caractères par une transformation de type rotation isométrique des points de l'image, le calcul des moments du premier et du second ordre peut être utilisé pour calculer l'angle de rotation.
- la *squelettisation* sert à obtenir une épaisseur égale à 1 du trait d'écriture et à se ramener ainsi à une écriture linéaire. Le squelette doit préserver la forme, la connexité, la topologie, les extrémités du tracé, et ne doit pas introduire d'éléments parasites.
- la *normalisation* permet de ramener les images de caractères à des tailles standard. Cette phase peut être indispensable pour certains systèmes comme

[1] sauf par celui qui l'a conçue

les réseaux neuronaux et les modèles de fusion que nous avons construits (voir la section 2.8).

L'étape principale d'un système d'OCR est d'extraire de l'image de caractère des caractéristiques sur lesquelles la reconnaissance est fondée. Les caractéristiques utilisées peuvent être différentes suivant la technique utilisée pour la phase de reconnaissance. Globalement les caractéristiques se divisent en trois types : *locales*, *globales* et *structurelles* [Hildebrandt et Liu(1993)] .

- les caractéristiques globales cherchent à représenter la forme générale d'un caractère et elles sont calculées sur l'image complète. Ces caractéristiques sont par exemple : les histogrammes de projections, les densités de points, les intersections avec des droites.
- les caractéristiques locales sont calculées au niveau d'un pixel et explorent la partie d'image autour de ce pixel. Elles peuvent aussi avoir à explorer des parties éloignées du pixel courant. La valeur même du pixel est un exemple simple de caractéristique locale que nous utiliserons dans le cas de la modélisation markovienne.
- les caractéristiques structurelles reflètent la structure du caractère (lettre ou chiffre) : présence de boucles fermées ou ouvertes, présence de jonctions caractéristiques entre segments.

1.4 Différentes approches de la reconnaissance

La reconnaissance est une tâche difficile à réaliser selon l'application demandée. C'est pour cela que les chercheurs ont développé plusieurs approches. On distingue l'approche statistique et l'approche structurelle. Une différence essentielle réside dans la représentation de la forme : vecteur de caractéristiques dans l'approche statistique, agencement de primitives pour l'approche structurelle. Nous allons rappeler quelques unes de ces approches couramment utilisées en les classant en 4 groupes : structurelle, statistique, neuronale et stochastique. Cette classification comprend une part d'arbitraire car celle-ci n'est pas unique. Nous nous limitons ici à la présentation de méthodes dites à apprentissage supervisé c-a-d. où la classe des observations est connue lors de l'apprentissage.

1.4.1 Approche structurelle

Les méthodes structurelles se basent sur la structure physique des caractères. Elles cherchent à décomposer le caractère en primitives et à décrire leurs relations.

Les primitives sont de type topologique comme un arc, une boucle, etc, et une relation peut être la position relative d'une primitive par rapport à une autre. Généralement le nombre de primitives est assez limité et leur enchaînement peut se décrire par un ensemble de règles d'assemblages. Dans cette approche on distingue plusieurs méthodes :

- **méthodes syntaxiques**

  Ces méthodes sont directement issues de la théorie des langages formels. Elles se basent sur une grammaire formelle. Chaque caractère est représenté par une phrase dans un langage où le vocabulaire est constitué de primitives. La reconnaissance consiste à déterminer si la phrase de la description du caractère peut être générée par la grammaire. Cette méthode a été utilisée par Ramesh [Ramesh(1989)] et Baptista [Baptista et Kulkarani(1988)]. L'inconvénient de ces méthodes est l'absence d'algorithmes efficaces pour l'inférence grammaticale directe.

- **comparaison des graphes**

  Cette méthode consiste à construire un graphe où les nœuds contiennent les primitives et les liens entre ces primitives. Ainsi la reconnaissance consiste à faire une mise en correspondance entre ce graphe et d'autres graphes représentant des caractères de référence construits lors de la phase d'apprentissage. Cette méthode a été utilisée par [Baird(1987), Lebourgeois(1991), Dargenton(1994)] et elle donne des résultats acceptables.

- **comparaison de chaînes**

  Dans ce cas, les caractères sont représentés par des chaînes de primitives. La méthode consiste à mesurer la similitude entre les chaînes du caractère à reconnaître et un modèle de référence par calcul de distance [Belaïd et Belaïd(1992)].

1.4.2 Classifieur bayésien

L'application des méthodes statistiques bayésiennes à la reconnaissance de forme a été formalisée par Chow [Chow(1965)]. Nous supposons que le problème de représentation des caractères admet un modèle probabiliste. Cette méthode définit l'appartenance d'un caractère à une classe avec un minimum d'erreur et évalue le risque de la décision à prendre pour un caractère donné. Nous cherchons donc la classe qui maximise la probabilité d'appartenance parmi l'ensemble des classes. L'intérêt de cette méthode est qu'elle repose sur des bases mathématiques[2] très bien définies.

[2]statistique : Règle de Bayes, Indépendance conditionnelle, $\cdots$

1.4.3 Méthodes des k plus proches voisins (KPPV)

Ces méthodes consistent, étant donné un point x de $\mathbb{R}^m$ représentant le caractère à reconnaître, à déterminer la classe de chacun des k points les plus proches de x parmi l'ensemble des caractères d'apprentissage et à retenir pour la décision la classe la plus représentée [Devijver(1987)]. Si k= 1, x est donc attribué à la classe de son plus proche voisin. De plus ces méthodes s'inscrivent dans le cadre des méthodes non bayésiennes et non paramétriques. Les méthodes des KPPV sont lentes en phase de décision puisqu'il faut calculer un nombre très important de distances. Nous remarquons que nous pouvons utiliser plusieurs types de distances parmis lesquelles la distance tangente qui utilise une approximation de Taylor du premier ordre [Beheim(2001)]. Pour cette distance il existe deux formes : la distance tangente unilatérale et la distance tangente bilatérale.
Pour résoudre le problème du temps de calcul des chercheurs ont proposé des variantes sub-optimales nécessitant moins de calculs[3].

1.4.4 Méthodes de séparation linéaire et non linéaire

Ces méthodes statistiques sont basées sur la définition des fonctions permettant de séparer partiellement ou totalement des classes représentées par les vecteurs paramètres de leurs échantillons [Tou et C.Gonzalez(1974)]. Ainsi on peut distinguer deux catégories de fonctions :

- fonctions de discrimination linéaires [Cornuéjols et Miclet(2002)], dans ce cas on parle d'un hyperplan séparateur (surface). Donc le problème consiste en la recherche du meilleur hyperplan permettant de discriminer les classes. Dans le cas de non séparabilité des classes, on passe dans un espace de plus grande dimension où l'on cherche un séparateur linéaire : ce sont les méthodes **SVM** (Support Vector Machines) [Vapnik(1998)].
- fonctions de discrimination non linéaires : fonctions linéaires par morceaux, séparation par des quadriques [Belaïd et Belaïd(1992)].

1.4.5 Méthodes connexionnistes

L'objectif est d'améliorer les capacités de la classification en utilisant des modèles aux composants fortement connectés. Un modèle connexionniste est un réseau dont les nœuds sont interconnectés par des liens pondérés. Ainsi, dans un réseau

[3]réduire l'espace de représentation, division de l'espace en cellules,etc ...

connexionniste, l'information est traitée par un grand nombre de processeurs élémentaires, chacun étant relié à un nombre important d'autres processeurs. En général le processeur est un *neurone formel* qui est un automate possédant n entrées réelles, son traitement consiste à effectuer à sa sortie le résultat d'une fonction de seuillage[4] de la somme pondérée de ses entrées. Vu que la plupart des modèles reposent sur le neurone formel, on parle souvent des *réseaux de neurones*. On peut dire aussi que les réseaux de neurones (RN) [Milgram et Schwenk(1996)] sont considérés comme des discriminateurs linéaires complexes. La complexité de la discrimination fonctionnelle que peut effectuer un réseau de neurones entraîne des conséquences :

- adaptation à des surfaces de séparation complexes.
- adoption d'une méthode d'apprentissage.

Malgré la capacité des RN d'implanter des techniques discriminantes très efficaces [LeCun *et al.*(2001)], et leurs performances de classement, la durée d'apprentissage peut être très longue.

1.4.6 Méthodes stochastiques

L'approche stochastique consiste à utiliser la modélisation par un processus stochastique [5] en prenant en compte la variabilité des caractères. Ainsi le caractère est considéré comme un signal continu observable dans le temps à différents endroits constituant des états d'observations. La reconnaissance consiste à comparer un échantillon de caractère de test à un modèle donné. Les paramètres des modèles[6] sont calculés à l'aide des probabilités calculées de manière plus fine par apprentissage. Chaque modèle décrit ces états à l'aide des probabilités de *transition* d'état à état et de la loi de probabilité d'observation par état. Ainsi la comparaison consiste à chercher dans ce graphe d'états le chemin le plus probable correspondant à une suite d'éléments observés dans la chaîne d'entrée. Ces méthodes sont robustes et fiables du fait de l'existence de bons algorithmes d'apprentissage.

1.4.6.1 Modèles de Markov cachés

Un modèle de Markov caché (HMM) [Rabiner(1989)]est défini par les données suivantes :

- un ensemble d'états
- un ensemble de symboles observables dans chaque état

[4] cette fonction est une fonction de Heavyside ou plus généralement de type sigmoïde

[5] Une collection de variables aléatoires (v.a.) $X = \{X(t), t \in [1, T]\}$ est appelée un processus stochastique

[6] classes, ou formes de référence

- une matrice des probabilités de transitions
- une matrice des probabilités d'observation
- un ensemble de densités de probabilité initiale.

Ce modèle utilise essentiellement des données unidimensionnelles, ce qui a permis leur application directe en traitement de la parole, et ensuite pour l'OCR. On peut distinguer deux types de HMMs :

- les HMMs discrets qui modélisent les observations avec des variables discrètes [Grandidier *et al.*(2000), Elms(1996)].
- les HMMs semi-continus qui modélisent les observations avec un mélange de gaussiennes [Gilloux(1994a)].

L'inconvénient est que les HMMs sont mono-dimensionnels alors que les images de caractères sont en dimension 2, ce qui donne des résultats parfois peu satisfaisants pour une application directe des HMMs.

1.4.6.2 Réseaux Bayésiens statiques et dynamiques

Un réseau bayésien statique (ou BN) [Pearl(1988)] est un mariage entre la théorie des graphes et la théorie des probabilités. Ainsi un réseau bayésien est constitué d'un graphe *acyclique orienté* dont les nœuds sont des variables aléatoires qui peuvent avoir un nombre discret d'états possibles ou dont les valeurs sont continues selon une loi continue, et d'un ensemble de distributions locales de probabilité qui sont les paramètres du réseau. Pour chaque nœud on dispose d'une table de probabilités conditionnelles.

Récemment les réseaux bayésiens sont appliqués dans le domaine de l'écrit. Nous pouvons citer les travaux de Cho dans lesquels il modélise les caractères on-line par des réseaux bayésiens : les variables qui représentent les positions des points caractéristiques, sont toutes observées. Souafi dans sa thèse [Souafi(2002)] utilise un réseau bayésien classifieur pour la classification automatique de documents (apprentissage des structures des BNs en utilisant la méthode de programmation génétique). Enfin les réseaux bayésiens ont été utilisés pour la vérification automatique des signatures [Xiao et Leedham(2002)] (modélisation des dépendances entre composantes de la signature).

Les réseaux bayésiens dynamiques (DBN) sont une extension des réseaux bayésiens statistiques et sont également appliqués à la reconnaissance automatique de la parole [Zweig(1998)]. Daoudi dans [Daoudi *et al.*(2002)] propose une approche multi-bandes où le réseau bayésien permet une interaction entre sous-bandes du signal de parole.

1.4.6.3 Approche 2D

L'aspect 2D des images des caractères a incité un certain nombre de chercheurs à trouver une autre modélisation markovienne. Les chercheurs ont essayé une technique d'amélioration des chaînes de Markov ou l'utilisation de champs de Markov avec des hypothèses simplificatrices.

- **modèles de Markov pseudo-2D** (Planar-HMM) [Saon(1997), Gilloux(1994b)]
  Ce sont des HMMs où la probabilité d'observation dans chaque état selon une direction, est donnée par un HMM secondaire. Les états du HMM principal sont nommés *super-états*. Ce sont des chaînes de Markov associées à l'autre direction comme le montre la figure Fig.1.1. L'inconvénient de ce modèle est

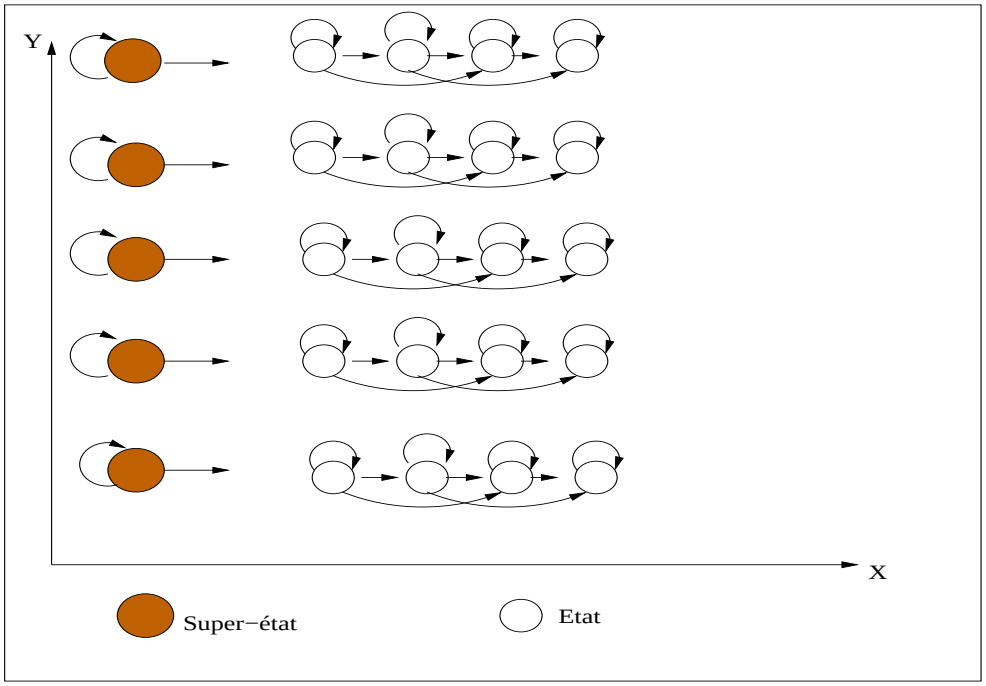


Fig. 1.1 – HMM-planaire d'après [Saon(1997)]

  la supposition que les lignes et les colonnes sont indépendantes contrairement à la réalité.

- **champs de Markov**
  Ils relèvent d'un modèle graphique non dirigé dont les nœuds sont des variables aléatoires et ils sont largement utilisés pour les traitements de l'image. Ils modélisent une structure 2D avec des dépendances locales (voisinages). Ces modèles sont peu utilisés en OCR et on distingue deux types suivant les contraintes de causalité : les réseaux de Markov [Gilloux(1994b)] et les champs de Markov unilatéraux [Park et Lee(1998)].

- **modèle NSHP-HMM** Saon [Saon(1997)] propose un modèle hybrique entre un champ de Markov observable et un HMM. Le NSHP-HMM est un modèle où les distributions du champ de Markov sont liées aux états d'un HMM observant les colonnes. Choisy [Choisy(2002)] a utilisé le NSHP-HMM pour normaliser les mots.

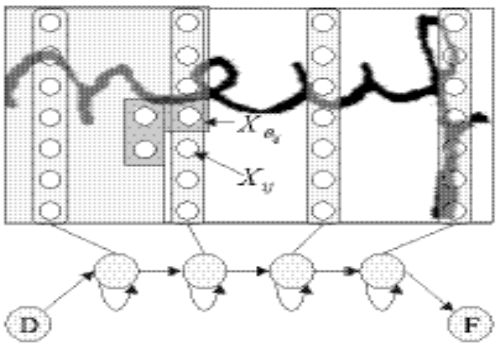


FIG. 1.2 – Application d'un champ de Markov sur le mot neuf d'après [Choisy(2002)] (NSHP-HMM)

- **Programmation dynamique 2D**

  C'est une approche par segmentation de l'image de caractère en régions. Cette segmentation proposée par Geoffrois [Geoffrois *et al.*(1998)] s'obtient par un algorithme de programmation dynamique 2D en optimisant la configuration globale par des optimisations partielles sur une des sous-régions (utilisation de champs de Markov, cliques et voisinages d'ordre 1). L'application de cette technique aux caractères est donnée par Chevalier [Chevalier *et al.*(2003)]

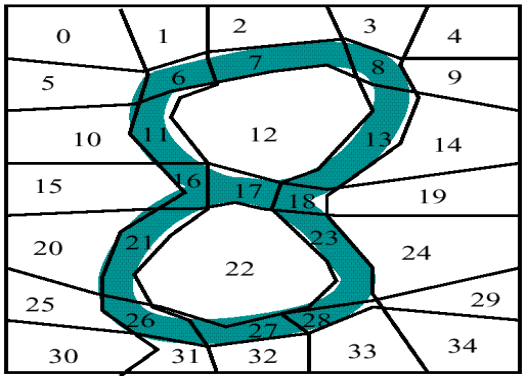


FIG. 1.3 – Segmentation en régions d'après [Chevalier *et al.*(2003)]

1.4.6.4 Conclusion

Nous avons introduit des définitions et des notions d'OCR. Ensuite nous avons exposé quelques approches pour la reconnaissance des caractères isolés. Dans les chapitres qui suivent nous allons exposer la modélisation des caractères imprimés dégradés à l'aide de chaînes de Markov cachées en remarquant sa limitation concernant la modélisation de l'aspect 2D de l'image des caractères. Ceci nous amènera a tenter de construire des modèles de couplage : fusion des données (balayage suivant l'axe diagonal de l'image du caractère) et fusion de scores (combinaison linéaire de deux HMMs construits séparément). Les résultats de la reconnaissance concernant ces modèles couplés nous ont encouragé à utiliser les *Réseaux Bayésiens*[7] dont les HMMs sont un cas particulier. Ces réseaux bayésiens permettent de modéliser les images de caractères en prenant en compte l'aspect 2D de l'image. Ces modèles sont plus adaptés à la réalité, puisqu'ils prennent en considération les corrélations entre les lignes et les colonnes de pixels de l'image des caractères (imprimés ou manuscrits).

[7]Nous allons étendre cette définition aux Réseaux Bayésiens Dynamiques

Chapitre 2

Modélisation des caractères par les Modèles de Markov Cachés (HMMs)

2.1 Introduction

Il existe plusieurs choix possibles quant à la modélisation et au traitement des caractères. Dans ce chapitre, nous allons définir une modélisation basée sur des propriétés statistiques à l'aide des modèles de Markov cachés (HMM). La supposition sous-jacente est que le modèle statistique de l'image peut être caractérisé comme un processus stochastique. Ce type de modélisation est largement utilisé dans le domaine de la reconnaissance de la parole. Un processus stochastique met en œuvre des modèles probabilistes spécifiques dans le but de gérer l'incertitude et le manque d'informations qui entachent les formes à reconnaître. La théorie des modèles de Markov cachés n'est pas nouvelle, elle est basée sur les publications de Baum [Baum et Petrie(1966), Baum et Egon(1967), Baum et Sell(1968), Baum et Sell(1970), Baum(1972)] dans les années 60 et 70. Les premières applications sont apparues à partir de 74 par Baker [Baker(1975)] pour la reconnaissance de la parole et s'appliquent encore largement à des systèmes de reconnaissance de la parole. Cependant les premières applications pour l'écriture sont apparues tardivement à partir de la fin des années 80 par Kordi [Kordi *et al.*(1987)]. Leur utilisation en reconnaissance de l'écrit s'explique naturellement du fait que les critères d'apprentissage de ces modèles s'inscrivent dans le cadre plus général de l'algorithme EM [Dempster *et al.*(1977)] dont le but est de réaliser un calcul itératif d'estimateurs de Maximum de Vraisemblance lorsque les observations (données) sont des données incomplètes du problème. On peut appliquer les HMMs sur n'importe quel type de signal. Il suffit d'avoir des primitives stables et de se munir d'une règle d'ordonnancement des observations. Notre objectif est d'appliquer la théorie des HMMs sur les

caractères imprimés dégradés.

2.2 Partie théorique

Dans cette partie, nous rappelons tout d'abord quelques définitions sur les variables aléatoires et les distributions des probabilités nécessaires à la compréhension des HMMs. Ensuite nous abordons les critères liés à la décision bayésienne et à l'estimation pendant l'apprentissage et l'algorithme EM. Puis nous abordons la théorie des HMMs.

2.2.1 Définitions et notations

Définition 1 Variable aléatoire
Soit (Ω, $\mathcal{F}$, P) un espace probabilisé.
Une variable aléatoire est une fonction $X : \Omega \longrightarrow S$ définie sur un espace de probabilité avec des valeurs dans un ensemble de réalisations (appelé ensemble d'états).
ie $\forall B \in S \qquad X^{-1}(B) \in \mathcal{F}$.

Définition 2 Variable aléatoire discrète
Une variable aléatoire est dite discrète si l'ensemble des réalisations est fini ou dénombrable. On note $P(X = x) = P(\{w \in \Omega \mid X(w) = x\})$ la probabilité d'une réalisation x de la variable aléatoire X (appelée aussi probabilité de masse).

Définition 3 Loi de probabilité d'une variable aléatoire
Soit X une variable aléatoire définie sur (Ω, $\mathcal{F}$, P) à valeur dans $\mathbb{R}$.
On appelle loi de probabilité P_X de X la mesure image de P par X .

Définition 4 *$f_w : \mathbb{R}^m \longrightarrow \mathbb{R}^+$, la distribution de probabilité de la classe w sur l'espace R^m. $f_w(x)$ notée aussi f(x,w) est la probabilité de trouver une forme de représentation x appartenant à la classe w.*

Définition 5 Fonction de densité de probabilité (FDP)
Soit $X : \Omega \longrightarrow \mathbb{R}$ une variable aléatoire réelle.
S'il existe une fonction f continue positive telle que :

$$P(X^{-1}([ab])) = \int_a^b f(x)dx \qquad \forall a, b \in \mathbb{R}$$

f s'appelle la fonction de densité de probabilité, elle satisfait à

$$\int_{\mathbb{R}} f(x)dx = 1$$

Définition 6 Indépendance conditionnelle
Soient X, Y, Z des variables aléatoires. On dit que X et Y sont indépendantes conditionnellement à Z si et seulement si

$$P(X, Y \mid Z) = P(X \mid Z) \times P(Y \mid Z)$$

Définition 7 Processus stochastique
Soit J un ensemble d'indices. Un processus stochastique est une famille $\mathbf{X} = \{X_j \mid j \in J\}$ *de variables aléatoires. Il est dit discret si les variables aléatoires sont en nombre fini ou dénombrable.*

Dans la suite, on utilise la notation suivante : $\mathbf{X} = \{X_j \mid j \in J\} \stackrel{Not}{==} X_J$

Définition 8 Processus markovien
Soit $\mathbf{X} = \{X_j \mid j \in J\}$ *un processus stochastique.* $\mathbf{X}$ *est dit markovien sur J si et seulement si on a l'indépendance conditionnelle de* X_A *et* X_B *sachant* X_C *pour certains sous-ensembles A, B, C de J.*

2.2.2 Décision Bayésienne

On s'intéresse à la décision qui garantit un taux d'erreur (risque) minimum. Notons $\mathbf{X}$ la variable aléatoire discrète dont les réalisations $\{ w_1, w_2, ..., w_n \}$ sont les classes possibles d'appartenance. On appelle $P(w_j) = P(\mathbf{X} = w_j)$ la probabilité a priori de la classe w_j. Supposons que la description y de la forme entrante soit la réalisation d'une variable aléatoire Y dont on connaît la FDP notée par $f(y \mid w_j)$ pour $j = 1, \cdots,$ n. En connaissant ces distributions et les probabilités a priori $P(w_j)$ il est possible de trouver la probabilité a posteriori d'une classe en utilisant la règle de **Bayes** :

$$P(w_j \mid y) = \frac{f(y \mid w_j) \times P(w_j)}{P(y)} \tag{2.1}$$

où

$P(y) = \sum_{i=1}^{n} f(y \mid w_i) \times P(w_i)$ est la probabilité a priori de la forme observée.

$P(w_j \mid y)$ représente la probabilité a posteriori de la classe w_j connaissant l'observation y.

Ainsi dans la règle de décision bayésienne, on peut faire intervenir les deux quantités $f(y \mid w_j)$ et $P(w_j)$. L'intérêt réside dans le fait que ces deux quantités sont plus facilement calculables.

2.2.3 Critères d'estimation pendant l'apprentissage

Nous avons vu précédemment que la décision bayésienne choisit pour y la classe w qui maximise :

$$f_w = f(y, w) = P(w) f(y \mid w)$$

Ainsi l'apprentissage consiste à déterminer $P(w)$ et $f(y \mid w)$.

■ **détermination de $P(w)$**

Cette probabilité est soit estimée facilement, soit connue a priori.

■ **détermination de $f(y \mid w)$**

Soit $f(y \mid w)$ la densité à estimer. On suppose que f dépend de m paramètres notés $\theta = (\theta_1, \theta_2, ..., \theta_m)$. Ainsi f est une fonction de θ et y, on le note par $f(y, \theta)$. Donc estimer la densité revient à estimer θ à partir de w.

2.2.3.1 Exemples d'estimation des paramètres : cas gaussien

Dans ce cas, les fonctions de densité de probabilité conditionnelle ont l'expression suivante :

$$f(y \mid w) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp[-\frac{1}{2}(y - M)^t \Sigma^{-1} (y - M)]$$

où

- d est la dimension de l'espace de représentation.
- M est le vecteur moyenne de la classe w.
- Σ est la matrice de covariance de la classe w.

Le principe de l'estimation paramétrique de la loi $f(y \mid w)$ supposée normale, est de trouver les paramètres M et Σ. Soit θ l'ensemble des paramètres. On considère un ensemble d'apprentissage constitué de n échantillons de la classe w. A ces échantillons on associe l'ensemble d'observation $Y = \{y_1, \cdots, y_n\}$. L'estimation suivant le critère du maximum de vraisemblance permet de trouver les paramètres de la loi qui

donnent à l'événement observé (ensemble Y) la probabilité maximale. En d'autres termes, les valeurs estimées $\hat{M}$ et $\hat{\Sigma}$ de M et Σ sont celles pour lesquelles on a la probabilité maximale de retrouver l'ensemble X par tirage aléatoire de n points (les tirages sont indépendants).
On pose $P(y \mid w) = P(y \mid \theta)$. On cherche θ tel que $P(y_1, \cdots, y_n \mid \theta)$ soit maximum. L'indépendance nous donne : $P(y_1, \cdots, y_n \mid \theta) = \prod_{i=1}^{n} P(y_i \mid \theta)$.
On considère la fonction $L(\theta)$ (log-vraisemblance) :

$$L(\theta) = \log[P(x_1, \cdots, y_n \mid \theta)] = \sum_{i=1}^{n} \log[P(y_i \mid \theta)]$$

$$L(\theta) \quad \text{maximum} \implies \nabla(L(\theta)) = 0$$

Ainsi d'après les calculs on trouve :

$$\begin{aligned} \hat{M} &= \frac{1}{n}\sum_{i=1}^{n} y_i \\ \hat{\Sigma} &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{M}) \times (y_i - \hat{M})^t \end{aligned} \tag{2.2}$$

2.2.3.2 Estimation par maximum de vraisemblance (MLE)

L'estimation par maximum de vraisemblance est l'une des principales méthodes utilisées pour l'apprentissage paramétrique supervisé. Soit $Y = \{y_1, y_2, ..., y_n\}$ un ensemble d'échantillons d'une même classe. En supposant que θ est connu et que tous ces échantillons ont été tirés indépendamment les uns des autres, on a :

$$f(Y \mid \theta) = \prod_{i=1}^{n} f(y_i \mid \theta) \tag{2.3}$$

La vraisemblance représente la probabilité que l'ensemble des échantillons X soit tiré en se basant sur les valeurs de θ. On appelle l'estimateur par maximum de vraisemblance le vecteur de paramètres :

$$\theta^{MLE} = \arg\max_{\theta} f(Y \mid \theta)$$

2.2.4 Algorithme EM

Cet algorithme est appliqué dans le contexte d'apprentissage non-supervisé où l'information de la catégorie (classe, état) associée à l'échantillon observé Y est

indisponible (cachée). Les données observables sont appelées *incomplètes* puisque la partie émettrice est absente. Le couple composé de données observables et de données cachées est appelé données *complètes*. Ainsi le but de cet algorithme, introduit par Dempster [Dempster *et al.*(1977)], est de réaliser un calcul itératif d'estimateurs du Maximum de vraisemblance lorsque les observations peuvent être considérées comme des données incomplètes du problème. Chaque itération de l'algorithme EM se décompose en deux étapes :

- Étape dite de l'espérance (Expectation step) ou étape E.
- Étape de maximisation (Maximisation step) ou étape M.

Supposons qu'il existe un espace X des données (variables) cachées associées à un espace Y de données incomplètes. Par exemple x peut être l'état émettant l'observation y. Pour $x \in X$ donné, le but est alors de maximiser la log-vraisemblance de la donnée observable y sachant θ.

Cette maximisation s'effectue de façon itérative, en procédant à chaque itération en deux temps nommés respectivement **étape E** et **étape M**.

▲ dans l'étape **E** , on part de l'ensemble de paramètres θ et on introduit la fonction auxiliaire :

$$\begin{aligned} Q(\theta, \theta') &= E(\log(f(x, y, \theta') \mid y, \theta) \\ &= \int \log(f(x, y, \theta') \times f(x \mid y, \theta) dx \end{aligned} \tag{2.4}$$

où θ' est l'ensemble de paramètres recherché. Le rôle de l'espérance a posteriori courante dans la fonction introduite est de résoudre le problème de la présence de la variable cachée. Dans le cas discret, on a

$$Q(\theta, \theta') = \sum_{x \in X} \frac{f(x, y \mid \theta)}{f(y \mid \theta)} \times \log f(x, y \mid \theta')$$

▲ dans l'étape **M** on souhaite déterminer l'ensemble θ^{k+1} tel que

$$\theta^{k+1} = \arg\max_{\theta} Q(\theta, \theta^k) \tag{2.5}$$

Cette étape réalise l'optimisation de la fonction auxiliaire établie dans l'étape E par rapport à l'ensemble des paramètres.

Ainsi on peut résumer l'algorithme de l'EM dans la figure Fig.2.1.

Algorithme EM

Etape E (Expectation) Calcul de $Q(\theta, \theta^k)$
Etape M (Maximisation) $\theta^{k+1} = \arg\max_\theta Q(\theta, \theta^k)$

FIG. 2.1 – Algorithme EM

2.2.5 Modèles de Markov Cachés

2.2.5.1 Introduction

Les modèles de Markov cachés (Hidden Markov Models ou HMMs) sont des modèles statistiques paramétriques de production du signal, largement utilisés en reconnaissance de la parole et plus tardivement en reconnaissance de l'écrit. Leur utilisation en reconnaissance de l'écrit s'explique du fait que les critères d'apprentissage de ces modèles s'inscrivent dans le cadre de l'algorithme EM. Les HMMs permettent de calculer la probabilité d'appartenance d'une forme à une classe en fonction du degré de distorsion subi par cette forme. Dans cette partie nous allons détailler la modélisation basée sur des propriétés statistiques des formes à l'aide des HMMs.

2.2.5.2 Définitions

Définition 9 Chaîne de Markov
Une chaîne de Markov discrète d'ordre n est un processus stochastique discret $\mathbf{X} = \{X_t \mid t = 1, ..., T\}$ *avec des variables aléatoires discrètes (les réalisations de ces variables sont appelées états), vérifiant la propriété de Markov :*

$$P(X_t = q_{i_t} \mid X_{t-1} = q_{i_{t-1}}, \cdots, X_1 = q_{i_1}) = P(X_t = q_{i_t} \mid X_{t-1} = q_{i_{t-1}}, \cdots, X_{t-n} = q_{i_{t-n}}) \quad \forall t \in [1, T] \tag{2.6}$$

où $Q = \{q_1, ..., q_n\}$ *représente l'ensemble des états.*

Dans la suite on n'utilise que les chaînes de Markov d'ordre 1.

Définition 10 Chaîne stationnaire
Une chaîne de Markov d'ordre un est stationnaire si pour tout t et k on a :

$$P(X_t = q_i \mid X_{t-1} = q_j) = P(X_{t+k} = q_i \mid X_{t+k-1} = q_j) \tag{2.7}$$

Dans ce cas, on définit une matrice de probabilité de transition A = (a_{ij}) telle que :

$$a_{ij} = P(X_t = q_j \mid X_{t-1} = q_i)$$

à un instant donné d'un processus quelconque.

Définition 11 *:*
Un modèle de Markov caché (HMM) est une chaîne de Markov cachée stationnaire où l'observation est une fonction probabiliste de l'état .

La figure Fig.2.2 représente le schéma d'un HMM.

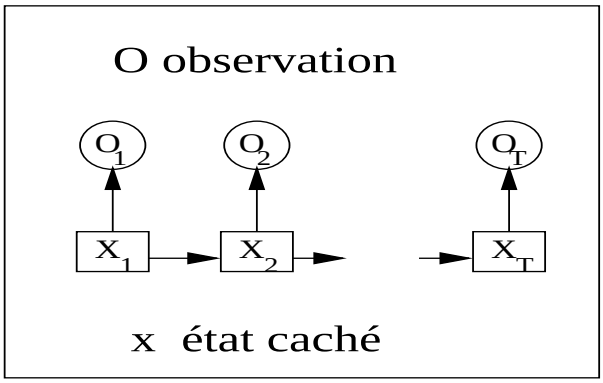


FIG. 2.2 – Modèle de Markov Caché

Remarque 1 *:*
Le modèle résultant est un processus doublement stochastique où la composante relative à la suite d'états est cachée. On notera par $\mathbf{O} = O_1O_2...O_T$ *la suite d'observations et par* $\mathbf{Q} = q_1q_2...q_T$ *la suite d'états (cachés)*

2.2.5.3 Modèle de Markov caché

Un HMM discret du premier ordre est défini par :

- N états représentés par $Q = \{q_1, q_2, ..., q_N\}$. On rappelle que l'on désigne un état au temps t par $q_t \in Q$.

- M symboles observés dans chaque état $V = \{o_1, o_2, ..., o_M\}$. On désigne une observation au temps t par $O_t \in V$.

- La matrice de probabilité de transition entre les états $A = (a_{ij})_{1 \leq i,j \leq N}$ avec

$$\begin{cases} a_{ij} = P(X_t = q_j \mid X_{t-1} = q_i) \quad \forall i, j \in [1, N] \\ a_{ij} \geqslant 0 \\ \sum_{j=1}^{N} a_{ij} = 1 \qquad \forall i \in [1, N] \end{cases}$$

- La matrice $B = (b_j(k))$ de probabilité d'observation des symboles dans chacun des états du modèle, $b_j(k)$ est la probabilité que l'on observe le symbole o_k alors que le modèle se trouve dans l'état j :

$$b_j(k) = P(o_k \mid X_t = q_j) \quad \text{avec} \quad k \in [1, M] \quad \text{et} \quad j \in [1, N]$$

De plus on a

$$b_j(k) \geq 0 \qquad \sum_{k=1}^{M} b_j(k) = 1 \quad \forall j \in [1, N]$$

- Ensemble π de densité de probabilité initiale $\Pi = (\pi_i)_{i=1,...,N}$

$$\begin{cases} \pi_i = P(X_0 = q_i) \quad \forall i \in [1, N] \\ \pi_i \geq 0 \\ \sum_{i=1}^{N} \pi_i = 1 \end{cases}$$

Par simplification , on désigne un HMM par le triplet $\lambda = (A, B, \Pi)$. Les fonctionnalités d'un HMM sont les évaluations de la probabilité d'émission d'une séquence d'observations O, l'apprentissage à partir d'un ensemble d'échantillons (observations) et la reconnaissance d'une suite d'observations.

2.2.5.4 Types de HMMs

Les principaux types de modèles de Markov cachés sont le modèle *ergodique* et le modèle *gauche-droite.*

- **modèle ergodique** c'est un modèle sans contraintes où toutes les transitions d'un état vers l'autre sont possibles (Fig.2.3) ie. $a_{ij} > 0 \quad \forall (i, j) \in [1, N]$.

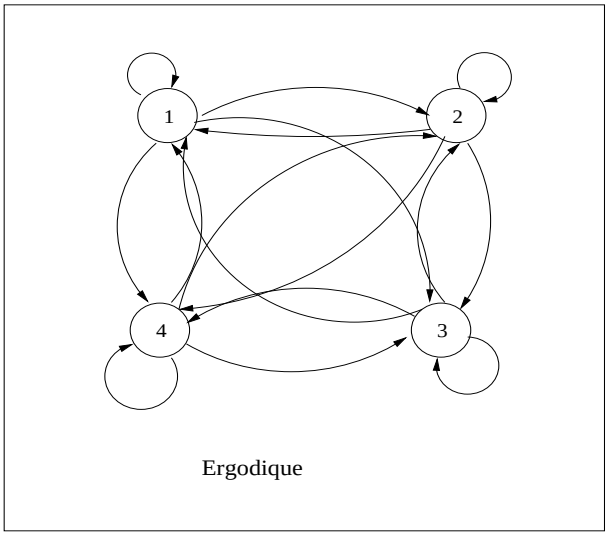


Fig. 2.3 – Modèle ergodique

- **modèle gauche-droite** c'est un modèle contenant des contraintes sur des transitions, c-a-d., il peut y avoir des interdictions de certaines transitions. Il existe deux sous-types de ce modèle : le modèle parallèle et le modèle séquentiel (Fig.2.4).

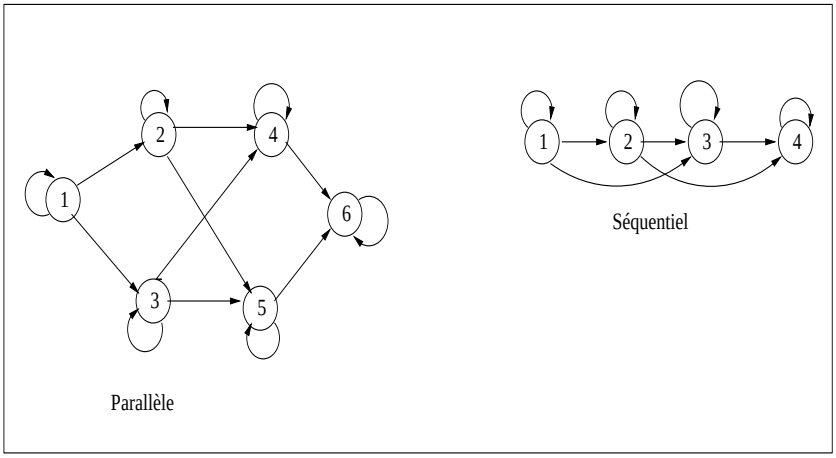


Fig. 2.4 – Modèles gauche-droite

On remarque que la connaissance de π est primordiale pour le choix de l'état de départ et que le modèle gauche-droite couvre la plupart des applications car dans ce cas on diminue le calcul des coefficients de la matrice de transition à estimer (moins de paramètres à estimer dans la partie de l'apprentissage).

2.2.5.5 Évaluation de la probabilité d'observation

Soient $O = o_1 o_2 ... o_T$ une suite d'observations et $Q = q_1 q_2 ... q_T$ la suite d'états associée.
La probabilité d'observation de O, étant donné le modèle λ (ou classe), est égale à la somme sur tous les chemins d'états possibles Q des probabilités conjointes de O et Q :

$$\begin{aligned} P(O \mid \lambda) &= \sum_{Q} P(O, Q \mid \lambda) \\ &= \sum_{Q} P(O \mid Q, \lambda) \times P(Q \mid \lambda) \\ &= \sum_{q_1, q_2 ... q_T} \pi_{q_1} b_{q_1}(o_1) \times a_{q_1 q_2} \times b_{q_2}(o_2) \times ... a_{q_{T-1} q_T} b_{q_T}(o_T) \end{aligned} \tag{2.8}$$

Ces calculs (ci-dessus) sont trop coûteux ($2TN^T$ opérations). Une évaluation optimale de cette probabilité est obtenue à l'aide de l'algorithme *Forward-Backward* de Baum-Welch [Rabiner(1988)]) qui permet de calculer $P(O \mid \lambda)$ de manière efficace. Seule la partie forward de cet algorithme est nécessaire pour ce calcul. La partie backward est nécessaire pour la phase d'optimisation des modèles.

2.2.5.6 Calcul $P(O \mid \lambda)$ à l'aide de fonctions Forward -Backward

Dans cette approche, l'observation peut se faire en deux temps :

- émission du début de l'observation $O(1 : t)$ en aboutissant à l'état q_i au temps t
- émission de la fin de l'observation $O(t+1 : T)$ sachant que l'on part de l'état q_i à l'instant t

Ainsi l'évaluation de l'observation est donnée par

$$P(O \mid \lambda) = \sum_{q_i} \alpha(t, q_i) \times \beta(t, q_i) \tag{2.9}$$

où

- $\alpha(t, q_i)$ est la probabilité d'émettre le début $O(1 : t)$ et d'aboutir à q_i à l'instant t.
- $\beta(t, q_i)$ est la probabilité d'émettre la fin $O(t+1 : T)$ sachant que l'on part de q_i à l'instant t.

Dans la suite on note $\alpha(t, q_i)$ par :

$$\alpha(t, q_i) \overset{Not}{==} \alpha_t(i) = P(o_1 o_2 \cdots o_t, q_t = i \mid \lambda)$$

Calcul de α

$\alpha_t(i)$ est calculée de manière récursive :

1. Initialisation :
$$\alpha_1(i) = \pi_i \times b_i(o_1) \qquad 1 \leq i \leq N$$

2. Récursion :
$$\alpha_t(i) = \left[\sum_{j=1}^{N} \alpha_{t-1}(j) \times a_{ji}\right] b_i(o_t) \qquad 1 \leq i \leq N \quad t = 2, \cdots T$$

3. Terminaison :
$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Calcul de β

$$\beta(t, q_i) \stackrel{Not}{=} \beta_t(i) = P(o_{t+1} o_{t+2} \cdots o_T \mid q_t = i, \lambda)$$

les β_t, β_{t+1} sont calculés de manière inductive :

1. Initialisation :
$$\beta_T(i) = 1 \qquad 1 \leq i \leq N$$

2. Induction :
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \times b_j(o_{t+1}) \times \beta_{t+1}(j) \qquad T - 1 \geq t \geq 1, \quad 1 \leq i \leq N$$

Remarque 2 *:*

- *Le calcul de α se fait avec t croissant tandis que le calcul de β se fait avec t décroissant, de plus le calcul de ces paramètres est de l'ordre N^2T.*
- *Le plus souvent on utilise les deux cas particuliers (t = 1) ou (t = T) ce qui donne :*
$$P(O \mid \lambda) = \sum_{q_i} \alpha_T(i) = \sum_{q_i} \pi_i \beta_1(i)$$

.

2.2.5.7 Apprentissage MLE

Il existe différents critères d'estimation en apprentissage, la plupart d'entre eux sont applicables aux HMMs. Le critère MLE est le plus utilisé. Le but de l'apprentissage MLE est de déterminer les paramètres (A,B, Π) qui maximisent le produit $\prod_{k=1}^{K} P(O^k \mid \lambda)$, où les $(O^k)_{k=1,K}$ sont les séquences d'observations des échantillons d'apprentissage.
Cela crée un problème lié à l'absence de critère d'optimisation globale et de méthode directe. Les solutions utilisées ne présentent que des optimisations locales. L'idée est donc d'utiliser l'algorithme de Baum-Welch ([Baum et Sell(1968)], [Baum(1972)]) basé sur le théorème de Baum-Welch ([Saon(1997)]) qui garantit l'atteinte d'un maximum local de la fonction de vraisemblance par ré-estimation des paramètres (A,B et Π) à partir d'un modèle initial.
On note :

- $P^k \overset{Not}{=} P(O^k \mid \lambda)$ la probabilité d'émission de l'échantillon O^k (qui peut être calculée par (2.9) pour le modèle courant).
- T^k sa longueur.
- $\alpha_t^k(i) = P(o_1^k o_2^k \cdots o_t^k, q_t = i \mid \lambda)$
- $\beta_t^k(i) = P(o_{t+1}^k o_{t+2}^k \cdots o_T^k \mid q_t = i, \lambda)$

On définit la fonction $\gamma_t^k(j)$ par :

$$\gamma_t^k(j) = P(q_t = j, O^k \mid \lambda) = \alpha_t^k(j) \times \beta_t^k(j)$$

C'est la probabilité de se trouver dans l'état j à l'instant t pour l'échantillon O^k.

les formules de ré-estimation

$$\overline{\pi_i} = \frac{1}{K} \sum_{k=1}^{K} \alpha_1^k(i) \times \beta_1^k(i) = \frac{1}{K} \sum_{k=1}^{K} \gamma_1^k \qquad 1 \leq i \leq N \tag{2.10}$$

$$\begin{aligned}\overline{a_{ij}} &= \frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\times a_{ij}\times b_j(O_{t+1})\times\beta_{t+1}^k(j)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\times\beta_t^k(i)} \\ &= \frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\times a_{ij}\times b_j(O_{t+1})\times\beta_{t+1}^k(j)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T_k-1}\gamma_t^k(i)} \qquad 1\le i,j\le N\end{aligned} \tag{2.11}$$

$$\begin{aligned}\overline{b_j(l)} &= \frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T_k}\alpha_t^k(i)\times\beta_t^k(j)\times\delta(o_t^k,v_l)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T_k}\alpha_t^k(j)\times\beta_t^k(j)} \\ &= \frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1,o_t^k=v_l}^{T_k}\gamma_t^k(j)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T_k}\gamma_t^k(j)} \qquad 1\le j\le N,\quad 1\le l\le M\end{aligned} \tag{2.12}$$

Les contraintes à respecter sont :

$$\begin{cases}\sum_{i=1}^{N}\overline{\pi_i}=1\\ \sum_{l=1}^{M}\overline{b_j(l)}=1\end{cases}$$

2.2.5.8 Reconnaissance

La reconnaissance peut être effectuée de deux façons différentes :

- Soit par la recherche du modèle discriminant dans le cas d'un modèle par classe.
- Soit par recherche du chemin optimal qui donnera la classe dans le cas d'un seul modèle pour toutes les classes.

Dans le premier cas on a une reconnaissance de type MAP par le calcul des probabilités d'émission de la forme en utilisant les modèles que l'on combine avec les

probabilités a priori des classes. La forme à reconnaître est affectée à la classe dont le modèle fournit la probabilité MAP

$$\lambda^* = \arg\max_{\lambda\in\Lambda} P(\lambda \mid O) = \arg\max_{\lambda\in\Lambda} P(O \mid \lambda)P(\lambda) \tag{2.13}$$

où Λ est l'ensemble des modèles et O la suite d'observations.

Dans le second cas , soient O une suite d'observations et λ un modèle donné. On s'intéresse à déterminer le meilleur chemin correspondant à l'observation c-a-d., trouver la meilleure suite d'états qui maximise $P(Q \mid O, \lambda)$, cette suite est appelée la suite d'états de *Viterbi*. Pour cela l'algorithme de Viterbi [Forney(1973)] définit $\delta_t(i)$ comme la probabilité du meilleur chemin amenant à l'état i à l'instant t en tenant compte les t premières observations :

$$\delta_t(i) = \max_{q_1,\cdots,q_{t-1}} P(q_1, q_2, \cdots, q_t = i, O_1, O_2, \cdots, O_t \mid \lambda) \tag{2.14}$$

Par induction, on calcule :

1. Initialisation :
$$\delta_1(j) = \pi_j \times b_j(O_1) \qquad 1 \leq j \leq N.$$
2. Récurrence croissante :
$$\delta_t(j) = \max_{1\leq i\leq N}[\delta_{t-1}(i) \times a_{ij}] \times b_j(O_t)$$
nous définissons la fonction $\psi_t(j)$ par :
$$\psi_t(j) = \arg\max_{1\leq i\leq N}[\delta_{t-1}(i) \times a_{ij}] \qquad 1 \leq j \leq N, t = 2 \cdots T.$$
3. Terminaison :
$$P(O, Q^* \mid \lambda) = \max_{1\leq i\leq N} \delta_T(i)$$
4. Séquence d'états
$$Q^* = \{q_t^*\}_{1\leq t\leq T}$$
obtenue par récurrence décroissante :
$$q_{t-1}^* = \psi_t(q_t^*) \qquad t = T, \cdots 2.$$

Nous verrons dans la suite que ces calculs se généralisent dans la partie des réseaux bayésiens et que dans nos applications, nous avons distingué deux cas des HMMs selon le type d'observation :

- **HMM semi-contimu** : les observations sont les lignes ou les colonnes de l'image de caractère
- **HMM discret** : les observations sont des valeurs obtenues à partir de la quantification vectorielle que nous détaillerons dans la suite.

2.3 Cas des HMMs discrets

L'idée est d'utiliser une approche de type de *quantification vectorielle (VQ)* des colonnes et des lignes de pixels de l'image de caractère [Elms(1996)]. Cet approche consiste à effectuer une quantification vectorielle au sens d'une distance d. Nous verrons plus loin dans les applications que les distances utilisées sont la distance euclidienne et la distance *SIHD* [Shift Invariant Hamming Distance] (distance de Hamming généralisée). L'objectif est la reconnaissance des caractères isolés ou des chiffres issus des documents dégradés en utilisant les HMMs avec des données quantifiées.

L'utilisation de la distance SIHD est justifiée du fait qu'elle donne une distance minimale par rapport à l'autre (euclidienne) qui calcule tout simplement le nombre de pixels différents entre deux vecteurs, alors que la distance SIHD est invariante par translation.

Pour cette approche nous avons besoin d'un dictionnaire qui est constitué d'un nombre fixe de vecteurs représentant les différents types de vecteurs présents dans la base d'apprentissage. Chacun de ces vecteurs possède un numéro d'ordre qui l'identifie dans le dictionnaire de façon unique. Ainsi on peut remplacer un vecteur (qui a le même nombre de composantes que les vecteurs du dictionnaire) par un scalaire, autrement dit l'indice du vecteur quantifié dans le dictionnaire. La figure Fig. 2.5 montre le cheminement de la quantification vectorielle.

Nous allons maintenant commencer par la définition de la distance SIHD.

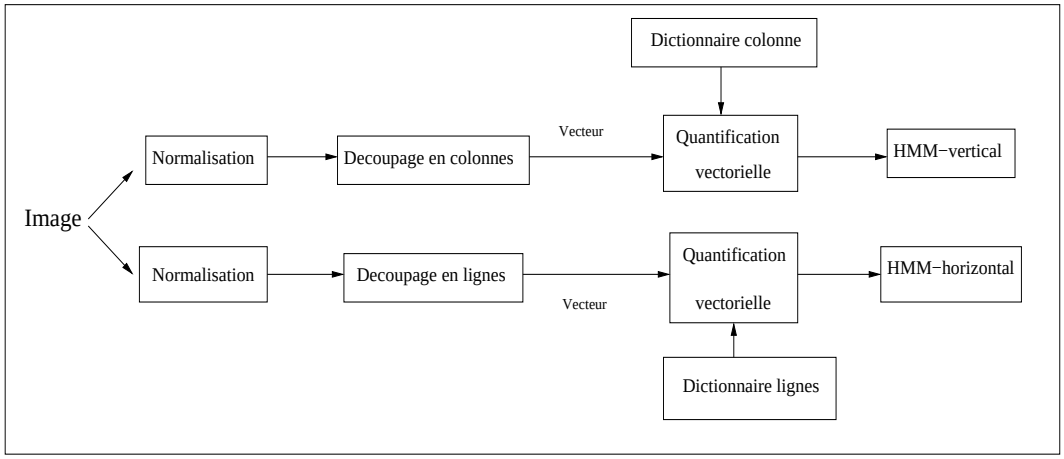


Fig. 2.5 – Chaîne de traitement

2.3.0.9 Distance SIHD

Soient deux vecteurs de pixels $\mathbf{a} = (a_1, \cdots, a_m)$ et $\mathbf{b} = (b_1, \cdots, b_n)$.
La dissimilitude entre ces deux vecteurs peut être mesurée par la distance euclidienne :

$$d^2 = \sum_{i=1}^{i_{max}} (a_i - b_i)^2$$

où

$$\begin{cases} i_{max} = \max(m, n) \\ a_i = 0 \qquad pour \quad i < 1, \quad i > m \\ b_i = 0 \qquad pour \quad i < 1, \quad i > n \end{cases}$$

(prolongement infini par des pixels de fond)
Si les vecteurs considérés sont binaires ie $a_i, b_i \in \{0, 1\}$, la distance est appelée *distance de Hamming* ie le nombre de pixels différents entre les deux vecteurs. Or deux vecteurs identiques mais décalés l'un par rapport à l'autre n'ont pas une distance nulle. D'où le besoin d'une autre distance pour résoudre ce phénomène. Elms [Elms *et al.*(1998)] introduit une distance invariante par translation (Shift Invariant Hamming Distance : SIHD). La distance SIHD est définie par :

$$SIHD(\mathbf{a}, \mathbf{b}) = \min_{\tau} \sum_{i=i_{min}}^{i_{max}} (a_i - b_{i+\tau})^2 \tag{2.15}$$

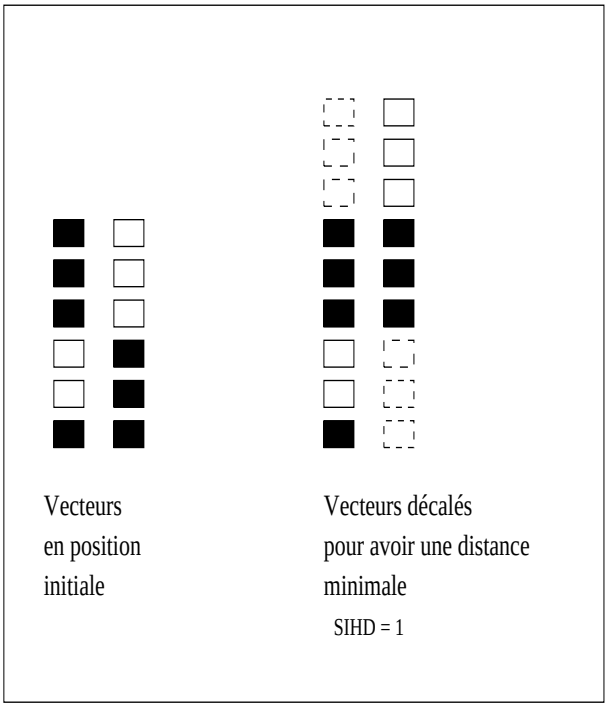
où

$i_{min} = -m$ et $i_{max} = m + n$

Notons que seuls les décalages compris entre $\tau_{min} = -n$ et $\tau_{max} = m$ peuvent donner la distance minimum. En effet en dehors de cet intervalle les pixels de non-fond (qui appartiennent aux caractères) ne se trouvent plus en face. Un exemple de calcul de cette distance est donné dans la figure Fig.2.6.

2.3.0.10 Construction du dictionnaire

Les vecteurs de dictionnaire (codebook) doivent représenter les différents types de vecteurs rencontrés de sorte que lors de la quantification vectorielle, il existe toujours un vecteur du dictionnaire peu différent du vecteur à quantifier. Pour choisir ces vecteurs on effectue une classification par la méthode des **k-moyennes**. D'après cette méthode, on obtient des classes de vecteurs et un représentant de chaque classe appelé *centre de classe*, qu'on prend comme vecteur de dictionnaire après binarisation (pour éviter les erreurs d'arrondis). Pour éviter des grossières erreurs d'arrondis au cours de la classification par les k-moyennes, les centres de classe sont



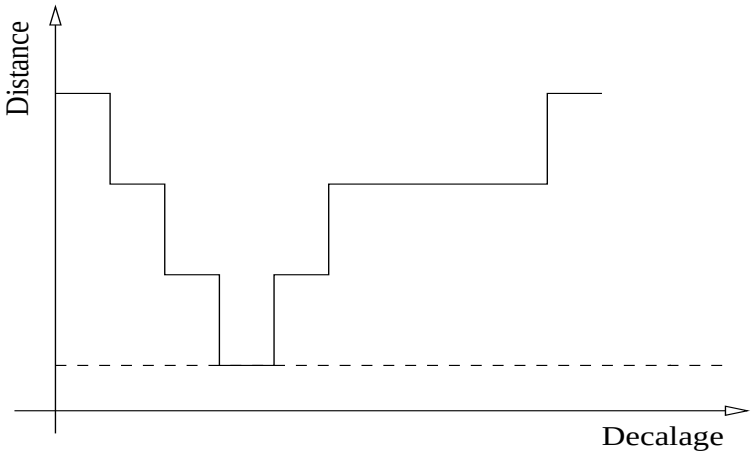


FIG. 2.6 – Exemple de calcul de la distance SIHD

donc considérés comme des vecteurs à composantes réelles comprises entre 0 et 1. Or les vecteurs de départ sont des vecteurs à composantes binaires. Il faut donc que le dictionnaire soit aussi composé de vecteurs binaires. Ainsi il est nécessaire de procéder à une binarisation des centres de classe pour obtenir le dictionnaire (il suffit de procéder à une binarisation par seuillage).

Remarque 3 .

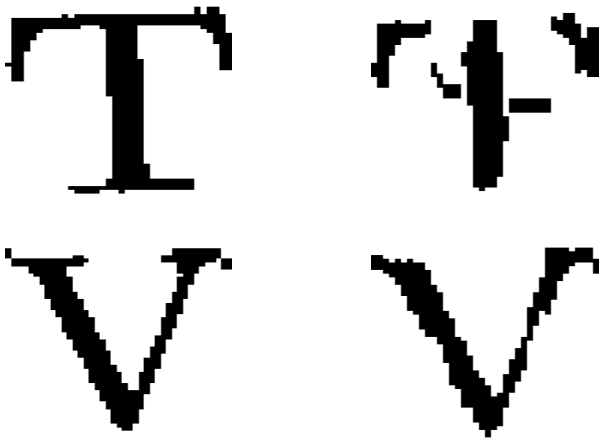
- *L'algorithme des k-moyennes est très sensible à l'initialisation et peut donner des résultats différents selon le choix de l'initialisation.*
- *Dans l'algorithme des k-moyennes nous avons testé les deux distances : euclidienne et la distance SIHD.*
- *Au cours de la construction du dictionnaire, nous avons interdit qu'une classe soit vide (les résultats sont meilleurs lorsqu'on utilise cette condition).*
- *La meilleure initialisation consiste à choisir les centres initiaux à partir des vecteurs de l'ensemble qu'on veut quantifier : nous avons choisi manuellement les vecteurs qui nous semblent les plus représentatifs de la diversité des vecteurs de l'ensemble.*

2.3.0.11 Quantification vectorielle

La quantification vectorielle consiste à remplacer un vecteur par l'un des vecteurs du dictionnaire le plus proche au sens d'une distance (dans notre cas c'est la distance SIHD ou la distance euclidienne). Dans le cas de la distance SIHD, on tient compte de la forme du vecteur et non de la position et donc on perd l'information sur la position des pixels. Pour pouvoir comparer, à des fins de visualisation, les caractères originaux avec ceux quantifiés, il faut donc placer de nouveau la colonne de pixels au bon endroit, et donc il faut stocker cette information avant la quantification. Cette information est contenue dans le centre de gravité (COG) du vecteur défini par

$$COG(a) = \frac{\sum_{i=1}^{M}(i+1) \times a_i}{\sum_{i=1}^{M} a_i} \tag{2.16}$$

On calcule donc le COG du vecteur avant quantification. Après quantification on décale le vecteur quantifié de la différence entre COG du vecteur quantifié et le COG du vecteur initial. Cette procédure permet d'opérer une quantification vectorielle respectueuse à la fois de la forme et de la position des pixels. Un exemple de résultat est présenté dans la figure 2.7 pour un dictionnaire de 16 éléments.



(a) Caractères originaux (b) Caractères quantifiés

Fig. 2.7 – Exemples de lettres quantifiées

2.4 Bases de données et pré-traitement

Pour tester les algorithmes HMMs et les HMMs couplés présentés plus loin, nous devons disposer de bases de données représentant les objets à étudier. Nous avons utilisé une base contenant des caractères imprimés et une base de chiffres manuscrits. Notre objectif est de mieux reconnaître les caractères isolés ou les chiffres présents dans des images.

2.4.1 UW English Document Image Database

Nous disposons de la base de données **UW English Document Image Database** [Askilsrud *et al.*(1993)] qui comporte :

- des pages de revues dégradées par photocopies successives. Ces pages sont indexées de sorte qu'il est possible de reconnaître le caractère ou le mot présent à un emplacement donné de l'image et inversement.
- des pages de caractères isolés dégradés artificiellement par le modèle de Baird [Baird(1993)] (appelé aussi **BLimd0**). C'est la partie que nous utiliserons dans la suite .

Cette partie de la base se compose de 94 symboles : A-Z, a-z, caractères spéciaux du code ASCII, et les images sont en format TIFF. De plus elle ne comporte que des caractères d'une seule police (Computer Modern Roman). En revanche, on y trouve plusieurs tailles de caractères. Pour chaque image on connaît :

- ▼ La hauteur et la largeur en pixels de la boîte englobant le caractère
- ▼ La police de caractère et la taille du caractère en points (unité typographique)

- ▼ Le symbole représenté (code ASCII du caractère)
- ▼ Les paramètres du modèle de dégradation de Baird qui a servi à la génération de cette image

Le modèle de Baird permet de générer une image résultat dégradée à partir d'une image d'un caractère idéal de taille 12, de très haute résolution, dont les valeurs de pixels sont continues entre [0.0, 1.0]. Nous donnons ci-dessous les paramètres de dégradation et entre crochets les valeurs de ces paramètres utilisées dans la base UW.

Les paramètres de dégradation sont :

- taille : représente la taille (en point) du caractère [4, 6, 8, 10, 12]
- résolution : modélise le résultat de la numérisation de caractères de différentes tailles [300 dpi]
- flou : les caractères sont filtrés par un filtre gaussien circulaire (écart type : paramètre de flou) [0.5 1 1.5 2 2.5].
- seuil de binarisation [0.2 0.25 0.3 0.35 0.4]
- sensibilité : modélise la sensibilité du capteur en ajoutant une valeur tirée aléatoirement à chaque pixel du caractère (loi normale, moyenne : 0.125, écart type : paramètre de sensibilité) [0 0.05 0.1].
- gigue : modélise l'irrégularité de la grille photo-réceptrice du capteur. Le centre d'un pixel est déplacé aléatoirement en x et en y (lois normales, moyenne : 0, écart type : gigue) [0]
- angle de rotation [-3 0 3]
- paramètres d'étirement (largeur, hauteur) : déformations qui modélisent celles apportées par la transmission par fax [0.9 1 1.11].
- déplacement vertical : modélise le décalage possible d'un caractère par rapport à la ligne de base du texte [-0.33 0 0.33].
- déplacement horizontal : modélise le déplacement horizontal du caractère dans l'image résultat [-0.33 0 0.33].

Suivant le choix des paramètres de dégradation, les caractères issus de cette base peuvent être très dégradés (pas lisibles) comme le montre la figure Fig.2.8.

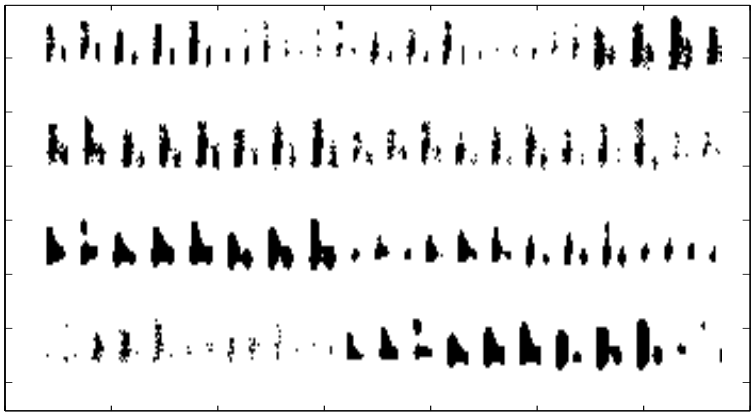


Fig. 2.8 – Exemple de caractères h dégradés

Les caractères de la base UW sont classés par classes de vecteur de paramètres. De notre côté nous avons décidé d'extraire des caractères de cette base en nous basant sur la lisibilité et non pas sur les coefficients de dégradation. Nous avons donc choisi dans la suite de l'étude un ensemble de caractères ayant subi différents taux de dégradation. Un exemple de caractères extraits de la base est dans la figure Fig.2.9.

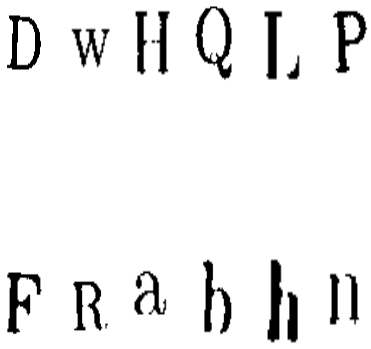


Fig. 2.9 – Exemples de caractères (base UW)

2.4.2 Base MNIST

La base MNIST est gratuite et elle est disponible sur le Web [LeCun(1998)][1]. Cette base de données se constitue des images de chiffres manuscrits extraites de la base de NIST (Special database1 (SD1) and Special database3 (SD3)).
La partie des images binaires concernant SD3 (collectée auprès des employés de

[1]http ://yann.lecun.com/exdb/mnist

Chiffres	nombres d'apprentissage	nombre de tests
0	1996	570
1	2283	688
2	1931	627
3	2078	597
4	1947	601
5	1777	550
6	1973	564
7	2095	611
8	1924	589
9	2016	623

Tab. 2.1 – Nombre de chiffres d'apprentissage et de test selon les classes (base MNIST)

bureau) est plus facile à reconnaître que SD1 (collectée auprès des étudiants). Ces images sont déjà pré-traitées :

- les images inscrites dans un carré normalisé de taille : 20×20.
- les caractères sont centrés, par contre on peut trouver des chiffres inclinés.
- l'ensemble d'apprentissage se compose de 60000 images de chiffres.
- l'ensemble de test est constitué de 10000 images.
- les images sont codées sur 8 bits.

Le tableau Tab.2.1 donne le nombre de chiffres d'apprentissage et de test par classe.

Dans cette base on trouve aussi les résultats d'évaluation de plusieurs techniques classiques de reconnaissance. Un exemple des chiffres de la base est donné dans la figure 2.10.

2.4.3 Traitement des données

2.4.3.1 Préparation des données

Avant de tester nos algorithmes, il faut commencer par la préparation des données.

♦ Pour la base UW :

On extrait les images de caractères de la base UW English Image Database en effectuant un recadrage, ensuite on les normalise car sinon les observations

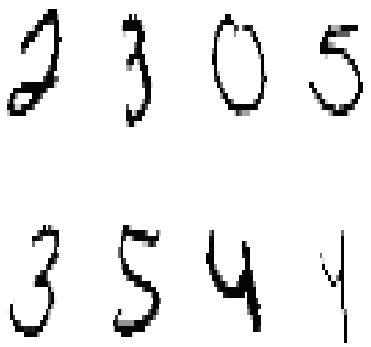


Fig. 2.10 – Exemples de chiffres (base MNIST)

n'auront pas les mêmes dimensions. Or le couplage des modèles HMMs (voir la partie de fusion) suppose que les séquences d'observations soient de même longueur.

Comme les plus grands caractères de la base ont des dimensions de 50x50 pixels, nous avons choisi la normalisation par homothétie de 50 pixels en ligne et 50 pixels en colonne. Ainsi on obtient des fichiers de caractères correspondant aux colonnes de pixels qu'on nomme fichiers colonnes, un exemple de ces fichiers normalisés est dans la Fig.2.11.

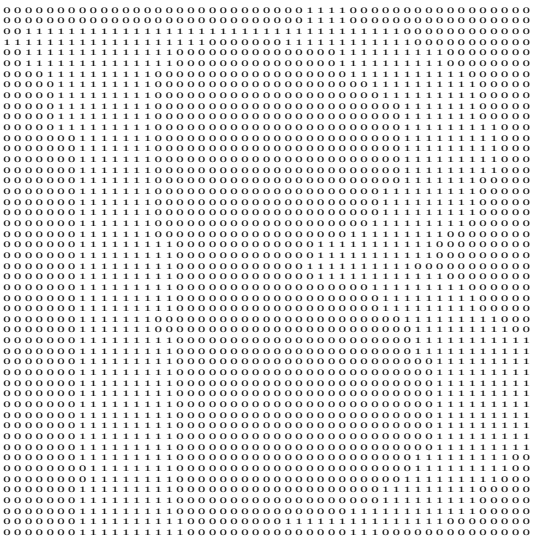


Fig. 2.11 – Fichier lignes correspondant à la Lettre B

De même on aura des fichiers des caractères correspondant aux lignes de pixels qu'on nomme fichiers lignes, un exemple est dans la Fig.2.12.

♦ Pour la base MNIST :

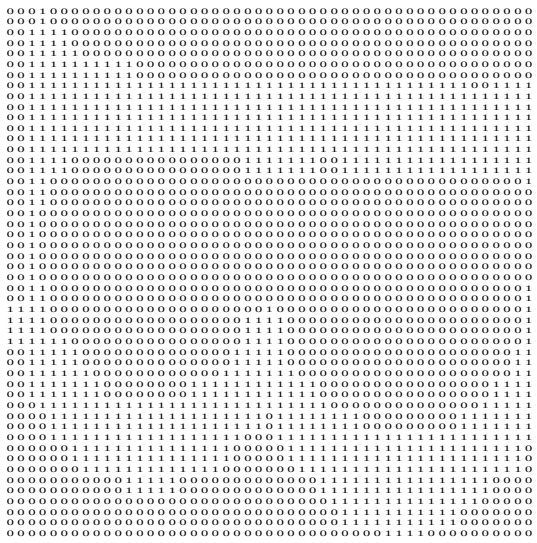


Fig. 2.12 – Fichier colonnes correspondant à la Lettre B

On extrait les images de chiffres de la base. On rappelle que les images sont déjà centrées dans un carré normalisé de 20×20. Nous les avons normalisées à 24×24. Dans la phase d'extraction des images, nous avons constaté que certains chiffres sont inclinés, donc nous avons effectué la procédure de redressement des chiffres (on cherche l'angle d'inclinaison par rapport à l'axe principal puis on fait une rotation). En ce qui concerne la construction des fichiers lignes et colonnes, nous avons respecté la même procédure qu'auparavant. Un exemple de chiffre après son traitement est dans la figure Fig.2.13.

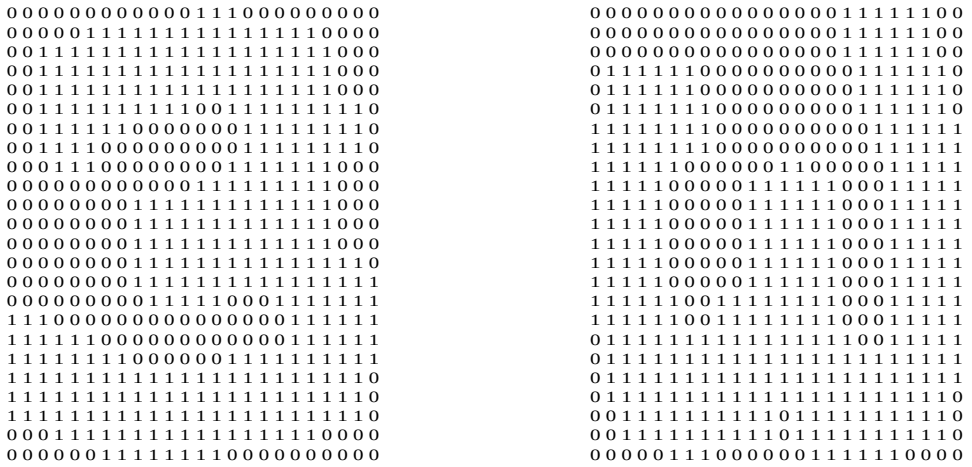


Fig. 2.13 – Fichiers lignes et colonnes correspondant au chiffre trois de la base MNIST

2.4.3.2 HTK et format htk

Pour l'expérimentation avec des HMMs, nous avons utilisé la boîte à outils de reconnaissance de la parole continue *HTK* [Young(2001)]. HTK permet de spécifier l'architecture des HMMs à utiliser (nombre d'états, nombre de gaussiennes, $\cdots$) la grammaire de la tâche de reconnaissance, etc. Afin d'utiliser HTK destiné par défaut à la reconnaissance de la parole, nous avons procédé à une série d'adaptations [Hallouli(2002)]. Le fonctionnement du package pour une tâche de reconnaissance de caractère (ou chiffre) isolé est le suivant :

1. **HInit** effectue une initialisation des paramètres des modèles par un algorithme de type k-moyennes (algorithme de Viterbi).
2. **HRest** effectue la ré-estimation Baum-Welch des paramètres des modèles.
3. **HVite** fait la reconnaissance Viterbi des fichiers de test en se basant sur la grammaire de la tâche.
4. **HResult** analyse les résultats en comparant les fichiers produits lors de la reconnaissance par HVite et calcule le taux de reconnaissance (matrice de confusion).

Avant d'utiliser le package HTK, nous avons converti nos données au format de celles utilisables par HTK, autrement dit en format htk. Ainsi un fichier de format htk se compose d'une séquence continue des observations précédée par un entête. Chaque observation est un vecteur ou bien un entier sur 2-octets ou bien un réel sur 4-octets. Les entiers sur 2-octets sont utilisés pour les formes comprimées comme décrit ci-dessous, l'en-tête de fichier de format htk est long de 12 octets et contient les données suivantes :

- nSamples : nombre d'échantillons dans le fichier (dans notre cas c'est le nombre de vecteurs de l'observation : 50)
- SamplPeriod : période d'échantillonnage (ne joue aucun rôle dans notre cas, par exemple on l'a fixé à 1000)
- SampSize : nombre d'octets par échantillonnage
- ParmKind : code indiquant la sorte d'observation (nombre entier sur 2-octets)

Ce dernier point ie ParmKind se compose d'un code de 6 octets représentant le genre de base de paramètre, plus les bits supplémentaires pour chacun des qualificatifs possibles. Nous avons utilisé les deux codes de paramètres de base suivants :

- 9 USER : l'utilisateur définit le genre de l'observation
- 10 DISCRETE : vecteur quantifié

Ainsi nous allons évaluer les résultats pour deux cas :

- **cas semi-continu** c'est le cas où les observations (colonnes, lignes) sont modélisées par des gaussiennes.
- **cas discret** c'est le cas où on utilise la quantification vectorielle des observations.

Dans l'annexe A, nous spécifions les fichiers de définitions des HMMs (cas semi-continu et discret), le réseau, le dictionnaire et le fonctionnement de HTK . Avant de commencer les expériences, nous rappelons que le type de HMM utilisé est le modèle gauche-droite comme le montre la figure Fig.2.14

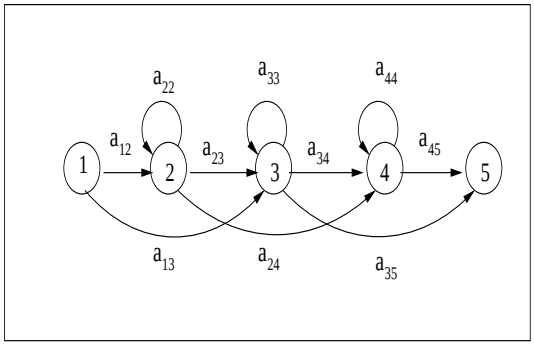


FIG. 2.14 – Modèle gauche-doite ($\Delta = 2$)

2.5 Résultats expérimentaux : cas semi-continu

Dans le cas semi-continu on procède par une modélisation des densités de probabilité par des gaussiennes. Le choix d'une seule gaussienne est justifié du fait que si on augmente le nombre de gaussiennes, on n'aura pas de résultat meilleur par rapport au cas mono-gaussienne. Par contre l'augmentation du nombre de gaussiennes implique un nombre important de paramètres à calculer.

Dans la suite on appelle :
HMM-vertical (respect *HMM-horizontal*) un modèle HMM qui prend pour séquence d'entrée les colonnes de pixels du caractère (respectivement les lignes de pixels) [Hallouli *et al.*(2002)] comme le montre la figure Fig.2.15.

2.5.1 Base UW ENGLISH

A partir de la base UW ENGLISH, nous avons constitué trois parties : apprentissage, validation et test (en effet nous avons besoin d'une base de validation pour régler différents paramètres de nos modèles)

- partie de l'apprentissage : nous avons 6240 exemples de caractères

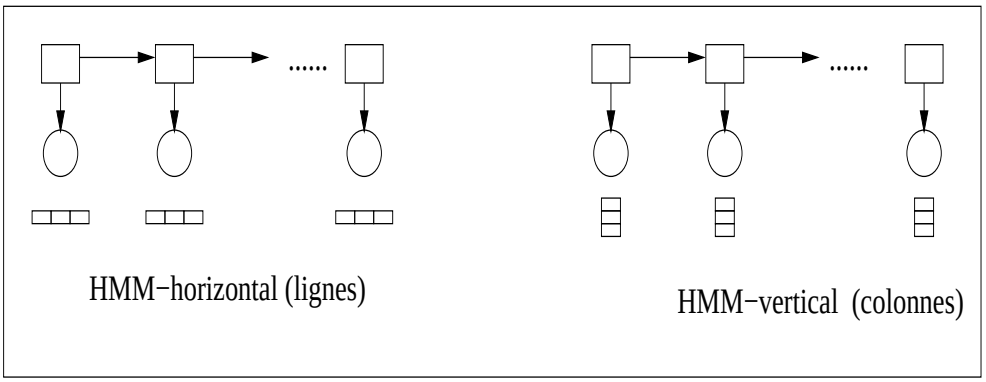


Fig. 2.15 – HMMs horizontal et vertical

– partie de validation : 2080 exemples de caractères
– partie de test : 2080 exemples de caractères

En effet avant de procéder aux tests, il faut régler tous les paramètres du modèle, en particulier le choix du nombre des états a une influence directe sur le taux de reconnaissance des caractères. Nous avons donc effectué sur la base de validation des tests concernant l'évolution du taux de reconnaissance par rapport au nombres des états en considérant le meilleur choix, les deux meilleurs choix et les trois meilleurs choix pour avoir le nombre d'états optimal. Après avoir fixé tous les paramètres nous avons procédé à une série de tests. Nous avons choisi le protocole suivant d'évaluation : sur un ensemble de caractères majuscules, un ensemble de caractères minuscules et un ensemble de caractères contenant des majuscules et minuscules mélangées.

2.5.1.1 Évaluation des résultats pour les caractères imprimés dégradés majuscules

Pour trouver le nombre optimal d'états cachés, nous avons procédé par une série de tests sur la base de validation concernant l'évolution du taux de reconnaissance par rapport au nombre d'états. Ainsi à partir de la figure Fig.2.16 qui montre l'évolution du taux de reconnaissance par rapport au nombre des états pour la base de validation concernant les majuscules, le nombre optimal des états cachés pour le HMM-vertical est de 13 états et de 11 états pour le HMM-horizontal. Nous remarquons que nous avons pris le même nombre d'états cachés pour tous les modèles. Cependant il se peut que le nombre d'états cachés soit différent suivant le modèle de caractère (par exemple entre I et W). Cette stratégie sera exploitée durant les expériences concernant les réseaux bayésiens.

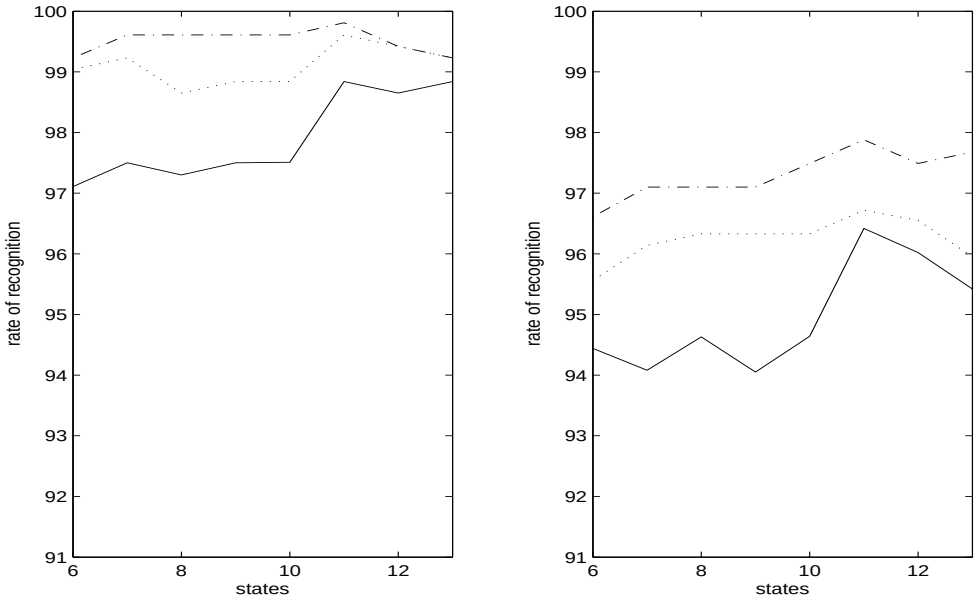


FIG. 2.16 – Taux de reconnaissance par rapport aux états pour les majuscules

Après avoir réglé tous les paramètres des modèles de caractères, nous passons à la phase de test. Ainsi les résultats concernant ce cas (Majuscules) sont donnés dans le tableau Tab.2.2.

	Valid-test	Nombre de caractères	Taux de reconnaissance	confusion
HMM-vertical	validation	2080	98.95 %	(I E) (L E)
	Test	2080	98.92 %	(I T)
HMM-Horizontal	validation	2080	96.45 %	(A E) (B H) (P D)
	Test	2080	96.32 %	(F E) (D B)

TAB. 2.2 – Résultats de reconnaissance pour les majuscules

Ainsi à partir de ces résultats, le HMM-vertical est meilleur que le HMM-horizontal, cela du fait que les vecteurs colonnes de l'image modélisent mieux le caractère. Nous remarquons qu'avec les vecteurs lignes, nous avons plus de confusions : par exemple entre les **B** et les **H**.

2.5.1.2 Évaluation des résultats pour les caractères imprimés dégradés minuscules

On procède de la même manière que dans le cas des majuscules. Nous commençons par la fixation de tous les paramètres des modèles. Ainsi nous cherchons le nombre optimal d'états cachés en utilisant la base de validation. Nous avons supposé que comme précédemment tous les modèles de caractères ont le même nombre d'états cachés. Cependant on pourrait varier le nombre d'états cachés selon le modèle de caractère. Ce dernier choix sera utilisé dans la partie des Réseaux Bayésiens. La figure Fig.2.17 qui représente l'évolution du taux de reconnaissance par rapport au nombre d'états, donne le nombre optimal d'états cachés concernant les minuscules. Donc le nombre optimal des états cachés est 13 états pour le HMM-vertical et de même, pour le HMM-horizontal le nombre optimal des états cachés est 13.

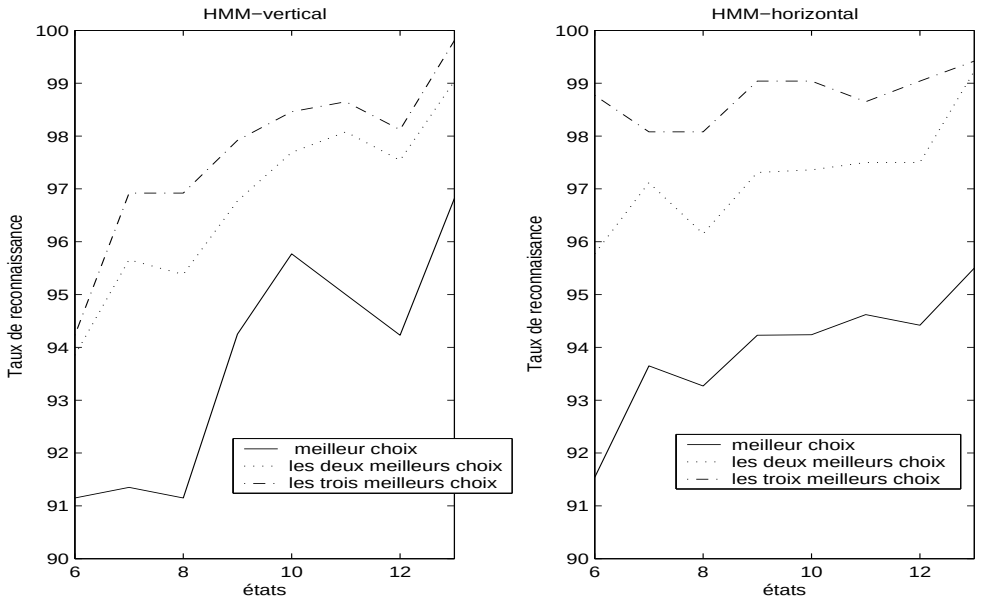


Fig. 2.17 – Taux de reconnaissance par rapport aux états pour les minuscules

Après avoir fixé tous les paramètres, nous passons à la phase de test. Ainsi les résultats numériques concernant ce cas (minuscules) sont dans le tableau Tab.2.3.

	Valid-test	Nombre de caractères	Taux de reconnaissance	confusion
HMM-vertical	Validation	2080	96.9 %	
	Test	2080	95.38 %	(l h) (n m) (v w)
HMM-horizontal	Validation	2080	94.49 %	(d g) (b k) (i j)
	Test	2080	93.26 %	(e c) (q d g) (t f)

TAB. 2.3 – Résultats de reconnaissance pour les minuscules

Dans ce cas, on constate que le HMM-vertical est meilleur que le HMM-horizontal et que les résultats concernant les minuscules sont moins bons que ceux des majuscules et que nous avons besoin de plus d'états. Cela est dû au fait que les caractères minuscules ont des formes plus complexes et nécessitent un ensemble de caractéristiques plus grand pour les décrire.

2.5.1.3 Évaluation des résultats pour les caractères imprimés dégradés : ensemble majuscules minuscules

De la même façon on teste la base de validation dans ce cas (ensemble majuscules minuscules) pour avoir le nombre optimal des états cachés. Nous remarquons que nous avons pris le même nombre d'états pour tous les modèles.
Ainsi à partir de la figure Fig.2.18, on constate que le nombre optimal est de 12 états pour le HMM-vertical et de 10 pour le HMM-horizontal.

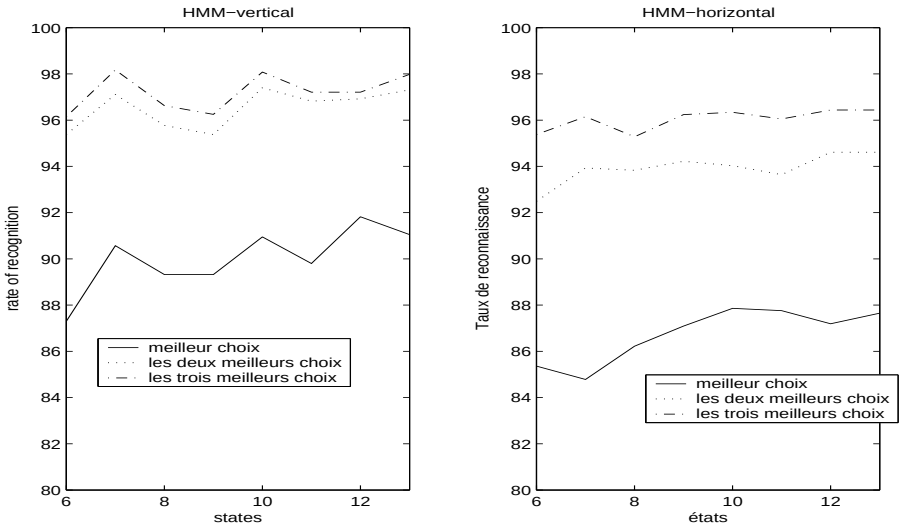


Fig. 2.18 – Taux de reconnaissance par rapport aux états pour l'ensemble majuscules-minuscules

Après avoir fixé le nombre d'états cachés, nous passons à la phase de test. Ainsi les résultats de test sont dans le Tab.2.4.

Dans ce cas, on constate que le HMM vertical est meilleur et que les taux de reconnaissance ont baissé puisque l'on trouve plus de confusion entre les classes de majuscules et de minuscules comme par exemple (O, o) et (X, x). Pour éviter ces confusions, on pourra utiliser le contexte des mots pour différencier les caractères qui ont les mêmes formes en majuscules et en minuscules. Nous remarquons aussi que le temps de calcul est plus important que dans les autres cas puisque nous avons 52 classes à construire et à utiliser pendant les tests.

2.5.1.4 Conclusion

Nos expériences montrent qu'une application directe des HMMs dans le cas semi-continu conduit à des résultats satisfaisants dans le cas où les deux parties majuscules et minuscules sont séparées. Les résultats obtenus permettent de comparer les modèles entre eux. Cependant ces résultats ont été obtenus en utilisant la base UW partiellement,et ne peuvent donc pas se comparer à d'autres méthodes puisqu'il faut avoir les mêmes données de test et d'apprentissage. C'est pour cela que nous avons également effectué des expériences sur la base MNIST qui est largement diffusée.
Nous remarquons que notre modélisation est jusqu'à maintenant mono-flux ie. nous avons utilisé uniquement les informations selon les colonnes ou bien les lignes. Ainsi notre objectif sera l'utilisation des informations simultanément selon les colonnes et

	Valid-test	Nombre de caractères	Taux de reconnaissance	confusion
HMM-vertical	Validation	4160	91.89 %	(F K) (G C) (I l) (o O)
	Test	4160	91.82 %	(H L) (S s) (x X) (R L) (Z z)
HMM-horizontal	Validation	4160	87.6 %	(G D) (I e L) (L b) (P p) (V v)
	Test	4160	86.66 %	(l I) (j J) (y V) (x X) (O o)

TAB. 2.4 – Résultats de reconnaissance pour l'ensemble majuscules-minuscules

les lignes. On peut déjà espérer un meilleur taux de reconnaissance si on introduit la stratégie où on considère le produit ou le maximum des deux vraisemblances des HMMs associées aux observations colonnes et lignes ou bien si on fait la moyenne des deux. Donc nous allons chercher par la suite une modélisation qui prend en compte l'aspect 2D de l'image des caractères en introduisant des stratégies qui prennent en considération cette remarque.

2.5.2 Application à la base de chiffres MNIST

En ce qui concerne cette base, nous avons construit trois parties à partir de l'ensemble d'apprentissage et de test fournis. Nous avons extrait de la base d'apprentissage une base de validation suivant le schéma suivant :

- partie d'apprentissage : cette partie est constituée de 40000 chiffres.
- partie de validation : constituée de 20000 chiffres.
- partie de test contenant 10000 chiffres.

Nous allons tester de nouveau nos algorithmes avec cette base de données (base de chiffres MNIST). Nous construisons de la même manière les deux HMMs : HMM-vertical et HMM-horizontal. Nous rappelons que pour trouver le nombre optimal des états cachés, nous devons tester notre base de validation pour avoir l'évolution du taux de reconnaissance par rapport au nombre d'états.
Ainsi à partir de la figure Fig.2.19 le nombre optimal d'états cachés pour le HMM-vertical est égal à 9 et pour le HMM-horizontal, le nombre optimal d'états est égal à 10.

	Val-test	Nombre de tests	taux de reconnaissance
HMM-vertical	validation	20000	97.45 %
	Test	10000	96.65 %
HMM horizontal	validation	20000	93.85 %
	Test	10000	93.68 %

Tab. 2.5 – Taux de reconnaissance par rapport aux états

Après avoir fixé tous les paramètres nous passons à la phase de test. Les résultats

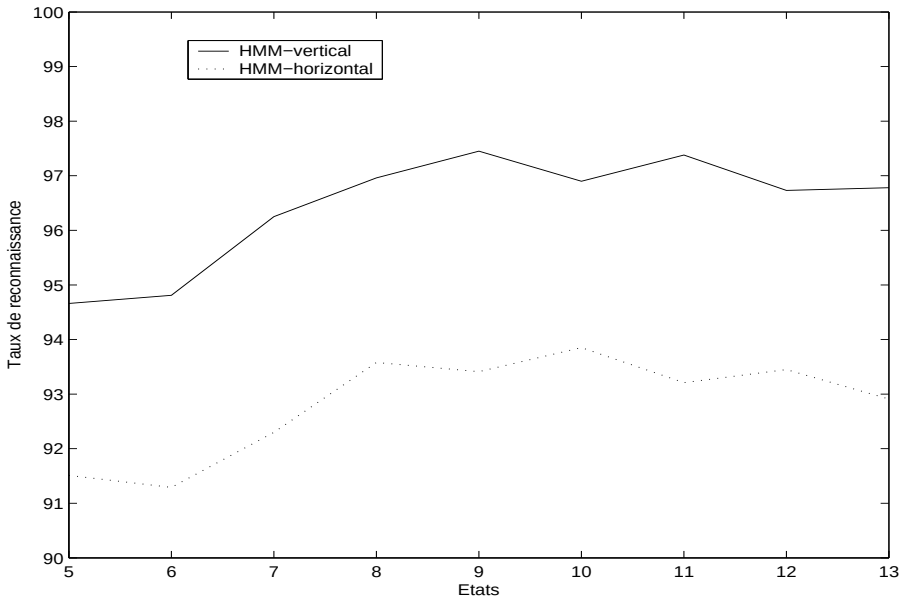


Fig. 2.19 – Taux de reconnaissance par rapport aux états pour les chiffres

associés sont donnés dans le tableau Tab.2.5.

À partir de ces résultats nous pouvons conclure de la même façon que pour les caractères imprimés, que le HMM-vertical est meilleur par rapport au HMM-horizontal. Cela est dû au fait que pendant l'écriture on se déplace plus verticalement que horizontalement et par exemple on a la confusion de 7 avec le 1 pour le modèle horizontal alors qu'on la trouve rarement pour le modèle vertical. Les modèles utilisant les réseaux de neurones [LeCun(1998), Simard *et al.*(2003)] atteignent sur la base MNIST entre 0.4% et 1.6% d'erreur, le taux le plus bas étant obtenu en augmentant la base d'apprentissage par des prototypes déformés artificiellement. Cependant le nombre de paramètres utilisés pour ces modèles est plus élevé et le temps de calcul est lent par rapport à notre méthode qui demande uniquement

quelques minutes.

2.6 Résultats expérimentaux : cas discret

Dans ce cas toutes les données sont quantifiées ie. chaque vecteur d'observation est remplacé par un indice d'un vecteur du dictionnaire. Nous avons cherché à voir l'influence de la taille du dictionnaire sur les taux de reconnaissance. D'autre part nous nous sommes posé la question sur la constitution du dictionnaire en fonction de notre protocole et aussi par le choix de la distance qui sera utilisée lors de la quantification vectorielle. Pour cela nous avons effectué nos expériences avec des dictionnaires de tailles différentes, et plusieurs protocoles de construction.
Nous avons essayé d'utiliser la distance euclidienne (ou de Hamming) lors de la quantification vectorielle, ce qui nous a conduit à des taux des reconnaissance moins satisfaisants. Cela nous a poussé à chercher une autre distance qui reste invariante pour certaines transformations et qui donne une distance minimale entre deux vecteurs. Nous avons opté pour le choix de la distance SIHD (distance Hamming généralisée) qui reste invariante par translation.
Nous construisons de la même manière que dans le cas semi-continu, les deux HMMs HMM-vertical et HMM-horizontal.

2.6.1 Évaluation des résultats pour les caractères imprimés dégradés majuscules

Nous allons chercher le nombre optimal d'états cachés à partir des expériences basées sur le dictionnaire de taille 16.
Pour obtenir le nombre optimal des états cachés, nous refaisons le même travail que dans le cas semi-continu. Nous notons ici aussi que nous avons pris le même nombre d'états pour tous les modèles.

Ainsi à partir de la figure Fig.2.20 qui représente l'évolution du taux de reconnaissance par rapport au nombre d'états, le nombre optimal pour le HMM-vertical est égal à 10 et pour le HMM-horizontal le nombre optimal des états est égal à 11. Ce sont ces nombres d'états optimaux pour les deux HMMs, qui seront choisis dans la suite même avec des dictionnaires de taille supérieure.

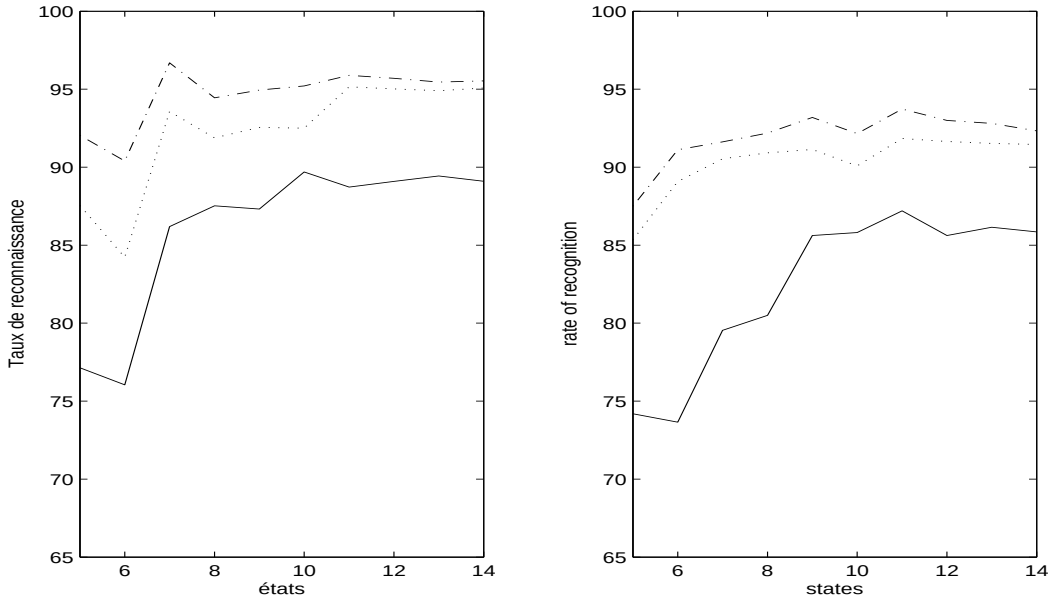


Fig. 2.20 – Taux de reconnaissance par rapport aux états pour les majuscules

Après avoir fixé le nombre d'états cachés en utilisant la base de validation, nous passons à la phase de test. Nous avons essayé de visualiser l'influence de la taille du dictionnaire sur les taux de reconnaissance. Pour cela nous avons commencé à évaluer les résultats avec un dictionnaire de taille de 16, ensuite nous avons augmenté la taille du dictionnaire à 30 vecteurs, puis nous avons augmenté la taille du dictionnaire à 40 vecteurs puis à 80 vecteurs et enfin à 120. Vu que les résultats se stabilisent à partir du dictionnaire de taille 80 vecteurs nous ne présentons que les résultats concernant les dictionnaires de taille 16, 30, 40 et 80. D'autre part cette taille limite de dictionnaire est à notre avis liée à la longueur de la séquence d'observation. Les résultats concernant les tests pour les différentes expériences selon la taille du dictionnaire sont donnés dans le tableau Tab.2.6.

À partir de ces résultats on peut conclure que :

- l'augmentation de la taille du dictionnaire jusqu'à une certaine taille (ici dans notre cas 80 vecteurs) implique directement une amélioration du taux de reconnaissance, ce qui est logique puisqu'on décrit de manière plus précise les données.
- le HMM-vertical donne toujours le meilleur taux de reconnaissance.
- l'augmentation de la taille du dictionnaire entraîne une augmentation de nombre des paramètres du modèle, donc un temps de calcul plus important.

	Taille du dictionnaire	Val-test	Taux de reconnaissance	confusion
HMM-vertical	16	validation	89.42 %	(C L) (G C) (Q O)
		Test	88.44 %	
	30	validation	92.3 %	(A w) (B H) (L D) (V W)
		Test	90.77 %	
	40	validation	92.83 %	(F T R) (B H) (N M) (L T) (V W)
		Test	90.96 %	
	80	validation	93.89 %	(L T) (B H) (T F) (W V)
		Test	92.46 %	
HMM-horizontal	16	validation	88.9 %	(C O) (D B) (P B) (L D) (Y N)
		Test	87.7 %	
	30	validation	90.33 %	(B E) (C O) (O Q)
		Test	89.19 %	(E F) (I J Z)
	40	validation	91.49 %	(M N) (V W) (P B)
		Test	89.74 %	(B H) (C O) (I L)
	80	validation	91.75 %	(G C) (B H) (E F)
		Test	89.84 %	

Tab. 2.6 – Résultats de reconnaissance pour les majuscules (cas discret)

2.6.2 Évaluation des résultats pour les caractères imprimés dégradés minuscules

De la même manière, nous cherchons dans ce cas le nombre optimal d'états cachés en utilisant la base de validation et un dictionnaire de taille 16.

Ainsi, on trouve que 11 est le nombre optimal d'états pour le HMM-vertical et aussi 11 est le nombre optimal d'états pour le HMM-horizontal comme le montre la figure Fig.2.21.

Après que nous ayons fixé le nombre optimal d'états cachés, nous passons à la phase

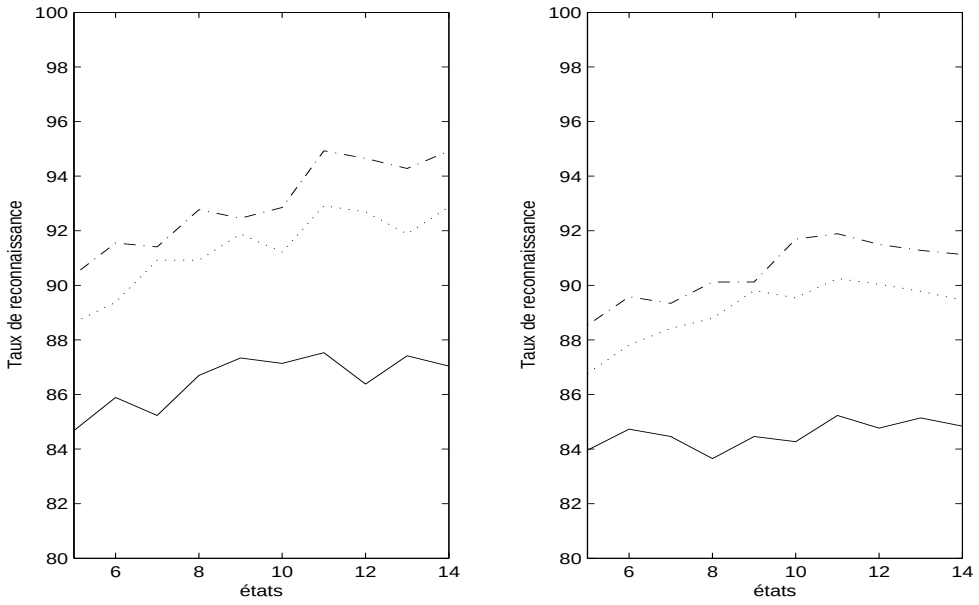


Fig. 2.21 – Taux de reconnaissance par rapport aux états pour les minuscules

de test. De la même façon nous faisons varier la taille du dictionnaire pour voir l'influence de la taille du dictionnaire sur les taux de reconnaissance. On remarque que ces taux se stabilisent à partir d'un dictionnaire de taille 80 vecteurs. Ainsi les résultats concernant les tests sont dans le tableau Tab.2.7.

Comme dans le cas des majuscules l'augmentation de la taille du dictionnaire jusqu'à une certaine taille (ici 80) implique une amélioration des taux de reconnaissance puisque le dictionnaire de taille 80 est celui qui représente le plus les différents vecteurs des images des caractères. Ces taux de reconnaissance sont cependant moins bons que ceux dans le cas semi-continu. Cela est justifié par la perte des informations au niveau de la quantification vectorielle. Nous remarquons aussi que certains résultats (cas d'un dictionnaire de taille 40) sont meilleurs pour les minuscules comparés aux majuscules, alors que nous avons le résultat inverse dans le cas semi-continu.

	Taille du dictionnaire	Val-test	Taux de reconnaissance	confusion
HMM-vertical	16	validation	91 %	(b h p) (d q)
		Test	87.53 %	(t f) (v w)
	30	validation	90.89 %	(c o) (i j l)
		Test	90.43 %	(n m) (v w)
	40	validation	92.04 %	(c o) (i j l)
		Test	91.55 %	(v w) (l t) (p b)
	80	validation	92.37 %	(c o) (i l)
		Test	91.87 %	(v w)
HMM-horizontal	16	validation	88.1 %	(i j) (o e c)
		Test	85.23 %	(z f) (q d) (v w)
	30	validation	88.39 %	(c o) (i j l) (l f)
		Test	88.03 %	(y x) (h n)
	40	validation	88.7 %	(b d) (c e) (i j)
		Test	88.17 %	(l f t) (d q) (y x)
	80	validation	88.9 %	(h n) (d q)
		Test	88.79 %	(c e) (f t)

TAB. 2.7 – Résultats de reconnaissance pour les minuscules (cas discret)

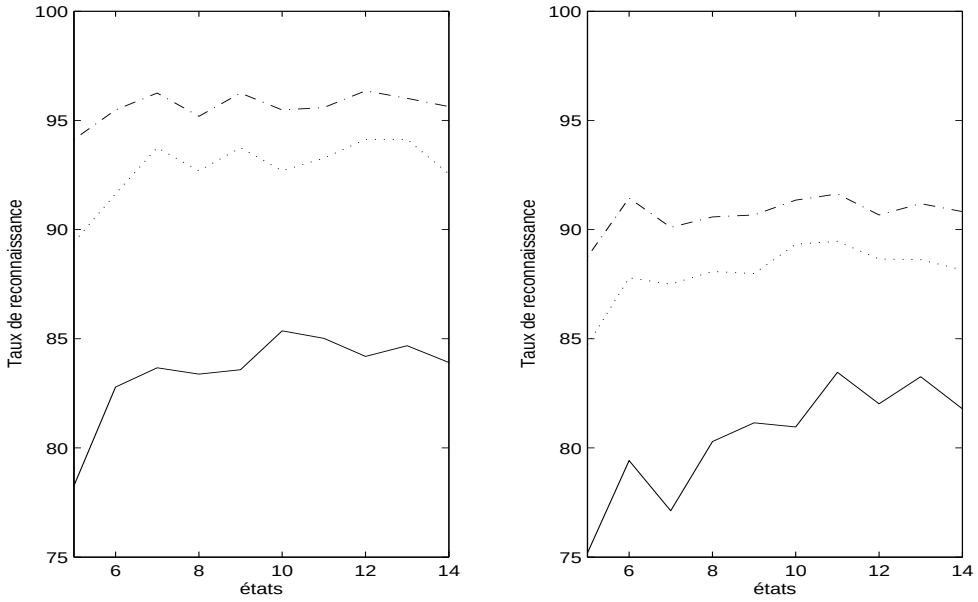


Fig. 2.22 – Taux de reconnaissance par rapport aux états pour l'ensemble majuscules-minuscules

Cela est dû à la construction du dictionnaire (classification par la méthode des k-moyennes). L'initialisation reste toujours un problème à résoudre dans ce type de méthode, et nous avons choisi manuellement comme vecteurs initiaux les vecteurs qui nous semblent les plus représentatifs de l'ensemble des vecteurs à quantifier.

2.6.3 Évaluation des résultats pour les caractères imprimés dégradés : ensemble majuscules minuscules

De même, nous cherchons le nombre optimal d'états cachés à partir des expériences sur la base de validation en faisant varier le nombre d'états cachés. Ainsi nous trouvons à partir de la figure Fig.2.22, 10 états comme nombre optimal des états cachés pour le HMM-vertical et 11 états comme le nombre optimal pour le HMM-horizontal.

Après avoir fixé le nombre optimal des états cachés nous passons à l'étape de test. De la même façon que précédemment nous varions la taille du dictionnaire pour avoir le meilleur taux de reconnaissance. Nous avons utilisé les tailles 16, 30 , 40 et 80 comme tailles des dictionnaires. Nous remarquons que pour la construction des dictionnaires il y a deux choix possibles :

– ou bien utiliser les deux dictionnaires existants pour les majuscules et les minuscules et les concaténer.

– ou bien construire un dictionnaire indépendant des deux déjà existants à partir de l'ensemble des données.

Nos expériences montrent que l'utilisation d'un dictionnaire constitué à partir de l'ensemble majuscules minuscules, est meilleur que la concaténation des deux construits séparément. Ceci est justifié par la dépendance des vecteurs des observations.

Les résultats de test en utilisant un dictionnaire constitué à partir de l'ensemble majuscules minuscules, sont donnés dans le tableau Tab.2.8

Ces résultats montrent qu'on a une baisse de taux de reconnaissance par rapport aux cas Majuscules et Minuscules séparées.

De plus l'augmentation de la taille du dictionnaire a une influence directe sur le taux de reconnaissance.

D'autre part les taux de reconnaissance sont moins bons que dans le cas semi-continu car lors de la quantification nous perdons des informations. Cette perte d'information est due à la construction du dictionnaire (trouver le dictionnaire optimal) et à la distance utilisée (trouver une autre distance que SIHD).

2.6.4 Base de chiffres MNIST

On recommence les mêmes expériences que précédemment ie. nous cherchons le nombre optimal des états cachés en utilisant la base de validation. Ensuite nous passons à l'étape de test. Ici nous nous limitons à un dictionnaire de taille 16. Ainsi avec 9 états cachés pour les deux HMM vertical et horizontal comme nombre optimal d'états, les résultats sont dans le Tableau Tab.2.9.

	Taille du dic	Val-test	Taux de reconnaissance	confusion
HMM-vertical	16	validation	86.21 %	(C c e) (O o) (I l)
		Test	85.36 %	(F E) (L r l) (V v) (L b) (S s) (X x)
	30	validation	88.11 %	(C e) (I i) (I r l)
		Test	86.85 %	(X x) (o O) (S s)
	40	validation	88.4 %	(l I) (J j)
		Test	87.23 %	(r T) (O o) (S s) (X x)
	80	validation	89.34 %	(O o) (l I)
		Test	88.06 %	(X x) (S s)
HMM-horizontal	16	validation	85.32 %	(L i) (i j l I)
		Test	83.25 %	(O O) (F E) (S s) (r t) (h n)
	30	validation	86.63 %	(L i) (i j l) (O o)
		Test	84.11 %	(S s) (h n) (r t)
	40	validation	86.91 %	(l I) (j J) (v W)
		Test	84.42 %	(O O) (v W) (Z z) (C o Q c)
	80	validation	87.9 %	"
		Test	85.03 %	"

Tab. 2.8 – Résultats de reconnaissance pour l'ensemble majuscules-minuscules (cas discret)

	Nombre de test	taux de reconnaissance
HMM-vertical	10000	78.84
HMM-horizontal	10000	73.46

TAB. 2.9 – Taux de reconnaissance pour les chiffres dans le cas discret

Les résultats obtenus montrent comme dans le cas des caractères imprimés, que la méthode de quantification vectorielle conduit à des résultats moins satisfaisants du fait que lors de la quantification on perd plus d'informations. De plus la taille du dictionnaire (ici 16) et le choix de ces vecteurs peuvent justifier cette dégradation du taux de reconnaissance par rapport au cas semi-continu. De même que pour les caractères imprimés, une taille du dictionnaire plus grande, par exemple de 32 éléments, serait clairement adaptée à la taille des images de chiffres (24×24).
Cependant le temps de calcul est réduit puisque l'on calcule moins de paramètres.

2.7 Conclusion

Nos expériences montrent que le modèle HMM-vertical est meilleur que le HMM-horizontal, du fait que les vecteurs colonnes des images de caractères sont plus discriminants que les vecteurs lignes pour la classification de caractères imprimés ou manuscrits et que les résultats associés aux majuscules sont meilleurs que les autres, cela est dû au fait que les majuscules ont des formes simples. Les résultats sont moins bons dans le cas de l'ensemble des majuscules minuscules puisque nous avons plus de confusion pour les caractères qui ont les mêmes formes. Ces confusions peuvent être enlevées par l'utilisation du contexte des mots.
Nous constatons aussi que le cas du HMM-semi continu est plus performant que le cas discret, puisque dans le cas discret on perd des informations concernant les images des caractères comme le montre la figure Fig.2.23.
Cette perte d'information se justifie par l'utilisation de la distance SIHD lors de la quantification vectorielle et aussi par le choix du dictionnaire. La distance SIHD est invariante par translation mais sensible aux dilatations. D'autre part le dictionnaire n'est pas forcement optimal car il dépend de l'initialisation. Cependant la taille du dictionnaire a une influence directe sur le taux de reconnaissance dans le cas de la quantification vectorielle.
De plus le temps de calcul est acceptable dans les deux cas semi-continu et discret.
Nous notons ici que nous avons aussi testé un HMM de type ergodique qui nous a conduit à des résultats moins bons que celui d'un HMM de type gauche-droite et que

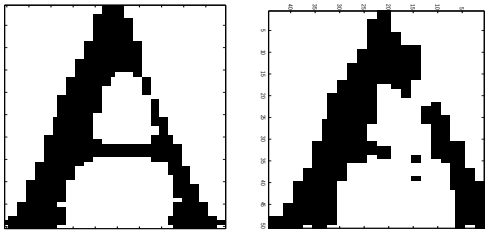


Fig. 2.23 – Lettre A avant et après la quantification (dictionnaire de taille 40) (quelque minutes suffisent pour tester un caractère test).

dans le cas semi-continu nous avons aussi testé l'influence du nombre de gaussiennes sur le taux de reconnaissance : les deux modèles mono-gaussien et multi-gaussien donnent des résultats semblables (c'est pour cela que nous avons opté pour le choix du modèle mono-gaussien).
Jusqu'à maintenant nous avons construit des modèles mono-dimensionnels[2]. Nous pouvons améliorer les taux de reconnaissance et nous espérons récupérer le manque d'information c-a-d., utiliser les informations selon les lignes et les colonnes simultanément ou plus précisément avoir une modélisation bidimensionnelle de l'image des caractères ou des chiffres, ce qui est plus proche de la réalité puisque les images sont en 2D.

2.8 Fusion des scores et fusion de données

À partir de la conclusion précédente sur les HMMs : HMM-vertical et HMM-horizontal sont des modèles mono-dimensionnels et les images de caractères sont bidimensionnelles, nous allons essayer de construire des modèles qui respectent l'aspect 2D des images de caractères et nous pouvons espérer une amélioration des taux de reconnaissance. Ainsi nous allons utiliser deux stratégies :

- une stratégie de fusion des scores : nous combinons linéairement les deux vraisemblances des deux classifieurs vertical et horizontal déja calculées séparément en se basant sur la remarque concernant la performance du classifieur vertical.
- une stratégie de fusion de données : nous allons construire un modèle qui prend comme séquence d'entrée les deux vecteurs ligne et colonne simultanément.

Ces modèles seront les premières tentatives d'une modélisation 2D.
Dans la suite nous allons nous limiter au cas semi-continu puisque ce dernier est

[2]les informations selon les colonnes ou les lignes

le cas le plus intéressant au niveau des scores de reconnaissance (pas de problème d'initialisation, pas de problème du choix d'une distance).

2.8.1 Fusion des scores

La stratégie de la fusion des scores est une méthode qui se base sur les probabilités d'observation issues de la reconnaissance, parmi les techniques existantes :

- prendre la moyenne des probabilités résultantes des deux HMMs (HMM-vertical et HMM-horizontal).
- faire le produit des probabilités résultantes des deux (HMM-vertical et HMM-horizontal).
- utiliser le max des probabilités résultantes des deux (HMM-vertical et HMM-horizontal).

Nous allons essayer de généraliser la méthode utilisée par Elms [Elms(1996)] qui consiste à utiliser uniquement le produit des probabilités résultantes du HMM-vertical et HMM-horizontal. En effet d'après nos résultats on constate que le HMM-vertical donne le meilleur taux de reconnaissance, ce qui nous a donné l'idée de favoriser les scores des HMM-verticaux en se basant sur :

$$L_f^{\alpha}(O \mid \lambda) = \alpha L_h(O \mid \lambda) + (1-\alpha) L_v(O \mid \lambda) \tag{2.17}$$

$$\widehat{\alpha} = \arg \max_{\alpha \in [\frac{1}{2},1]} \sum_{O \in B} \mathbb{1}\{\arg \max_{\lambda \in C} L_f^{\alpha}(O \mid \lambda) = C(O)\} \tag{2.18}$$

où $L_v(O \mid \lambda)$ (respect. $L_h(O \mid \lambda)$ est la log de vraisemblance résultant du HMM-vertical (respect. HMM-horizontal) pour l'observation O et le modèle λ, B est l'ensemble des observations de la base de validation, C ensemble des classes et $C(O)$ est la classe associée à l'observation O.
Nous allons justifier dans la suite la forme de la log vraisemblance écrite dans l'équation 2.17

Cette méthode consiste à combiner les deux vraisemblances résultantes des deux HMMs. On suppose que les vecteurs de l'observation, extraits du même classifieur, sont indépendants.
Le vecteur de l'observation est décomposé en deux parties o^v et o^h, les paramètres θ de la classe λ se decomposent en (θ^h, θ^v). La suite d'états optimal dans les deux directions est noté $\hat{x^h}$ et $\hat{x^v}$.

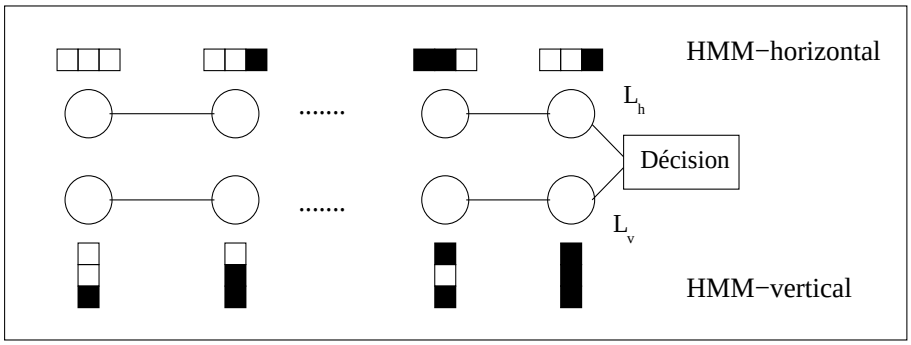


FIG. 2.24 – Fusion des scores

Ainsi on a :

$$\begin{aligned} P(O=o \mid \theta) &= P(O^h=o^h \mid \theta^h) \times P(O^v=o^v \mid \theta^v) \\ &= \frac{[P(O^h=o^h, X^h=\hat{x^h}(o^h) \mid \theta^h)]^\alpha}{Z^h} \\ &\times \frac{[P(O^v=o^v, X^v=\hat{x^v}(o^v) \mid \theta^v)]^\beta}{Z^v} \end{aligned} \tag{2.19}$$

avec

$\alpha, \beta \in \mathbb{R}$ et $\alpha + \beta = 1$.

les Z^v et Z^h sont des constantes de normalisation.

En prenant le logarithme de l'équation 2.19

$$L_f^\alpha(O \mid \lambda) = \alpha L_h(O \mid \lambda) + \beta L_v(O \mid \lambda) - \log(Z^h) - \log(Z^v) \tag{2.20}$$

Nous allons justifier le fait que pour un choix correct des paramètres α et β, Z^h et Z^v sont quasiment indépendants du modèle.

On suppose que l'observation est modélisée par une gaussienne. Sans perte de généralité, on suppose que cette gaussienne est mono-dimensionnelle

$$\mathcal{G}(\xi, \mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp(-\frac{(\xi-\mu_i)^2}{2\sigma_i^2}) \qquad \xi, \mu_i, \sigma_i \in \mathbb{R}$$

Ainsi la fonction Z^h est donnée par

$$\begin{aligned} Z^h &= \int_{o^h} [P(O^h=o^h, X^h=\hat{x^h}(o^h) \mid \theta^h)]^\alpha do^h \\ &\approx \int_{o^h} [\frac{1}{|X^h|} P(O^h=o^h \mid X^h=\hat{x^h}(o^h), \theta^h)]^\alpha do^h \\ &= \prod_{t=1}^{T_h} \int_{-\infty}^{+\infty} [\frac{1}{N} \mathcal{G}(\xi, \mu_i(\hat{x^h}(o^h)), \sigma_i(\hat{x^h}(o^h)))]^\alpha d\xi \end{aligned} \tag{2.21}$$

où T_h est la longueur de l'observation.
posons $\mathbf{Ad}_i(o)$ l'ensemble des observations menant vers l'état optimal $\hat{x^h}(\xi) = i$, $i = 1, \cdots, N$
donc

$$Z^h = \left[\sum_{i=1}^{N} \int_{\mathbf{Ad}_i(o)} \left[\frac{1}{N} \mathcal{G}(\xi, \mu_i, \sigma_i \right]^{\alpha} d\xi \right]^{T_h}$$

de plus supposons que les variances sont proches et que les moyennes μ_i sont classées par ordre croissant. Ainsi on a :

$$Z^h = (z^h)^{T_h}$$

avec

$$z^h \approx \frac{1}{(\sqrt{2\pi}N)^{\alpha}} \times \left[\sqrt{\frac{\pi}{2\alpha}} (\sigma_1^{1-\alpha} + \sigma_N^{1-\alpha}) + \sum_{i=1}^{N-1} \Phi_{\alpha}(A_i)(\sigma_i^{1-\alpha} + \sigma_{i+1}^{1-\alpha}) \right] \tag{2.22}$$

où

$$\Phi_{\alpha}(u) = \int_0^u \exp -\frac{\alpha \xi^2}{2} d\xi$$

$$A_i = \frac{\mu_{i+1} - \mu_i}{\sigma_{i+1} + \sigma_i}$$

D'où

- si $\alpha > 1$ alors $\lim_{\sigma \to 0} Z^h = \infty$ donc il y a une forte dépendance du modèle
- si $\alpha < 0$ alors Z^h est aussi sensible au modèle

Ainsi ces calculs justifient le choix de α et β qui doivent être entre 0 et 1 pour avoir une indépendance relative du modèle, et donc l'équation 2.20 est équivalente (relativement) à

$$L_f^{\alpha} = \alpha L_h + \beta L_v \tag{2.23}$$

De plus les valeurs de α et β ont une influence directe sur le processus de fusion, nous les avons choisies en favorisant les scores résultant du HMM-vertical puisque ce dernier donne le meilleur score de reconnaissance. Dans nos expériences, après plusieurs tests numériques, nous avons pris $\alpha = 0.25$ et $\beta = 0.75$.

2.8.2 Fusion de données

Dans ce cas nous avons utilisé un modèle qui regroupe les deux observations comme le montre la figure Fig 2.25. Donc pour ce cas notre modèle prend en considération simultanément les lignes et les colonnes. Le vecteur d'observation est constitué de la fusion du vecteur ligne et du vecteur colonne mais une seule variable d'état

contrôle les deux observations.
Ici le fait d'utiliser simultanément la ligne et la colonne peut être interprété comme un balayage suivant l'axe diagonal.

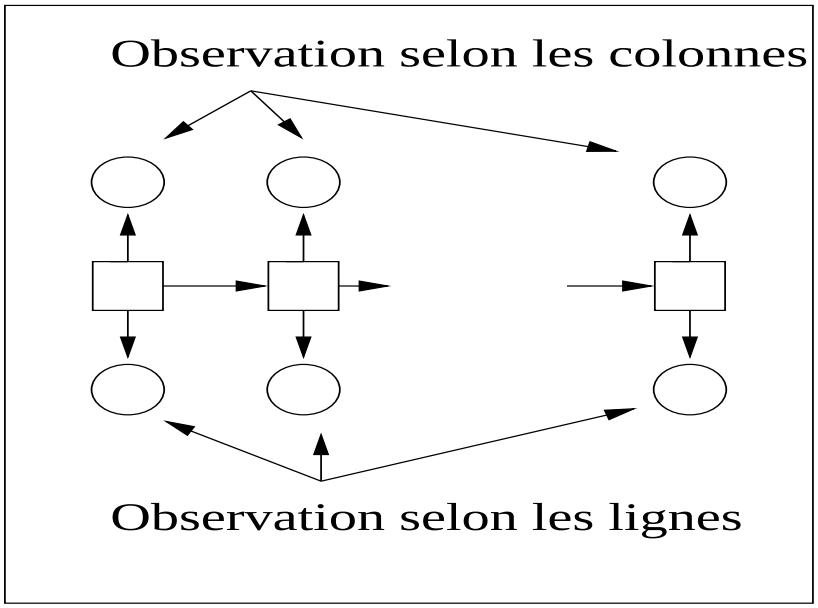


Fig. 2.25 – Modèle de fusion de données

Nous avons construit ce modèle en utilisant l'option multiflux (multistream) de HTK. Cette option construit une matrice de covariance en concaténant les matrices de covariance de chaque flux, ce qui donne une matrice diagonale par blocs. De plus le modèle de fusion de données a moins de paramètres que celui de la fusion des scores (une seule variable d'état pour les flux horizontal et vertical d'observation).

2.8.3 Résultats expérimentaux

Nous testons les deux méthodes : fusion de données et fusion des scores .

- pour la fusion des scores : nous construisons deux modèles résultant de deux HMMs, nous combinons ensuite linéairement les deux log-vraisemblances en se basant sur l'équation 2.23.
- la fusion des données : nous construisons un modèle qui prend en considération l'information selon la ligne et la colonne. Nous cherchons le nombre optimal des états cachés en faisant varier le nombre des états et en utilisant la base de validation. Ainsi nous avons trouvé que le nombre optimal des états cachés est égal à 12, et nous avons supposé que tous les modèles ont le même nombre d'états.

Les résultats des méthodes de fusion de données et des scores pour les caractères imprimés dégradés, ainsi qu'un rappel des résultats des HMMs de la même base, sont dans le tableau Tab.2.10. De même les résultats des méthodes de fusion de

	Taux de reconnaissance		
Ensemble des lettres	Majuscule	Minuscule	Majuscules-minuscules
Fusion de données	99.7	99.8	93.82
Fusion des scores	98.9	98.6	93.3
HMM-vertical	98.8	95.3	91.8
HMM-horizontal	96.4	93.2	86.6

TAB. 2.10 – Résultats de reconnaissance pour les différents modèles

données et des scores pour les chiffres (MNIST) sont dans le tableau Tab.2.11.
Tous ces résultats montrent que les deux modèles de couplage sont meilleurs que les

	Chiffres	Nombre d'états
Fusion de données	97.7	9
Fusion des scores	96.9	–
HMM-vertical	96.6	9
HMM-horizontal	93.6	9

TAB. 2.11 – Résultats de reconnaissance concernant les chiffres pour les différents modèles

modèles classiques et que le modèle de fusion de données est le meilleur du fait que pour ce modèle on élimine l'hypothèse de l'indépendance des classifieurs. Cependant le temps de calcul est plus important que dans le cas des HMMs simples.

2.9 Conclusion

Le modèle HMM-vertical est meilleur que le HMM-horizontal car les vecteurs colonnes des images de caractères sont plus discriminants que les vecteurs lignes pour la classification de lettres ou de chiffres manuscrits, ce qui n'est pas nécessairement le cas pour d'autres types d'écritures.
Les résultats associés aux minuscules sont moins bons. Cela est dû au fait que les caractères minuscules ont des formes complexes et nécessitent un ensemble de caractéristiques plus grand pour les décrire. Nous obtenons plus de confusion pour les caractères de même forme dans le cas où nous utilisons l'ensemble des majuscules

minuscules. Cette confusion peut être enlevée si nous utilisons le contexte des mots. De plus le cas discret donne des résultats moins bons que le cas semi-continu. Cela est dû à la perte de l'information lors de la quantification vectorielle. Cette perte est liée à la distance utilisée (SIHD) et à la construction du dictionnaire (classification par k-moyennes).

D'autre part les stratégies de fusion sont meilleures par rapport aux modèles classiques. Cela provient du fait que la stratégie de fusion prend en compte l'aspect 2D de l'image de caractère, ie. l'utilisation de l'information selon la colonne et la ligne, et que les HMM classiques sont mono-dimensionnels.

Enfin la stratégie de la fusion des données est meilleure que celle de fusion des scores car dans la fusion des scores on suppose que les deux classifieurs vertical et horizontal sont indépendants. De plus à partir de la visualisation des états cachés on peut déduire que le modèle de fusion de données est plus riche au niveau de l'information que les autres. Un exemple d'évolution des états par rapport aux observations est donné dans la figure Fig.2.26. Ce schéma d'évolution n'est pas cependant facile à interpréter car le nombre d'états varie selon les différents modèles utilisés. De plus le temps de calcul est plus important dans le cas de fusion que dans le cas des HMMs simples.

Cependant dans le cas de la fusion de données les observations sont supposées in-

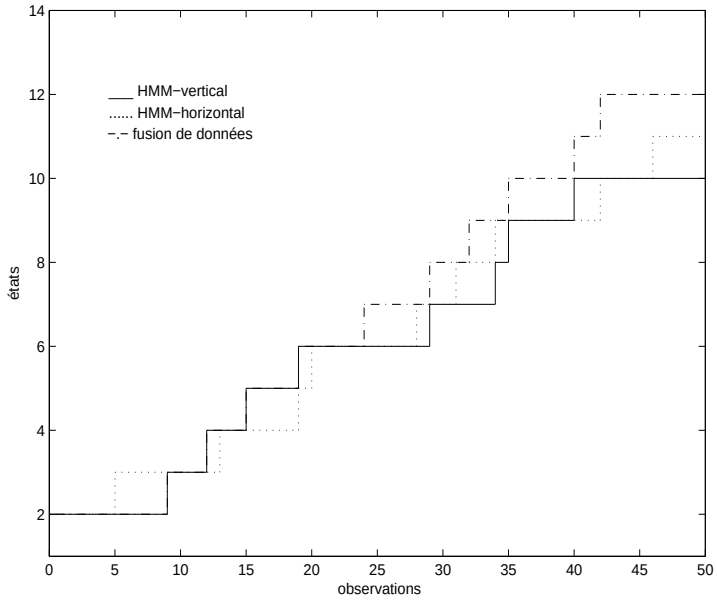


Fig. 2.26 – Évolution des états par rapport aux observations pour le HMM-vertical, le HMM-horizontal et la fusion de données pour le caractère **A** (modèle gauche-droite)

dépendantes entre elles ce qui n'est pas vrai dans la réalité. Ainsi tous ces résultats et ces discussions nous encouragent à chercher d'autres stratégies de fusion ou de couplage.

Chapitre 3

Réseaux Bayésiens

Les réseaux bayésiens sont des modèles représentant des connaissances incertaines sur des phénomènes complexes. Ils sont une union entre la théorie des probabilités et la théorie des graphes afin de donner des outils efficaces pour représenter une distribution de probabilités jointe sur un ensemble de variables aléatoires. La représentation de la connaissance se base sur la description, par des graphes, des relations de causalité existant entre les variables définissant le domaine d'étude. A chaque variable on associe une distribution de probabilités locale spécifiant une relation causale.
Ce chapitre est une synthèse sur les réseaux bayésiens : il présente les principales définitions et les principaux théorèmes et étapes de la construction des réseaux bayésiens. On présente d'abord le formalisme des réseaux bayésiens en utilisant les notions d'indépendance conditionnelle. Ensuite on présente l'algorithme de Jensen Lauritzen Olesen (JLO) servant comme moteur d'inférence exacte que nous utilisons pour nos expériences et on termine par la partie de l'apprentissage des paramètres avec des données complètes et incomplètes. Ce chapitre donne une idée globale sur les réseaux bayésiens, il aborde les points nécessaires pour comprendre ce domaine.

3.1 Introduction

Les réseaux bayésiens constituent une technique d'acquisition, de représentation et de manipulation de connaissance. Toute application mettant en œuvre des connaissances peut relever de l'utilisation des réseaux bayésiens, qu'il s'agisse de formaliser la connaissance d'experts, d'extraire la connaissance contenue dans des bases de données, ou d'utiliser le plus rationnellement possible l'un ou l'autre type de connaissance.
On utilise les réseaux bayésiens pour leur capacité d'effectuer des inférences dans un contexte d'incertitude, en quelque sorte comme alternative aux systèmes experts. On les utilise aussi pour leurs algorithmes d'apprentissage, comme alternative aux autres méthodes de modélisation quantitative, en les considérant comme des modèles de régression.
Les réseaux bayésiens sont le résultat de recherches effectuées dans les années 80, et sont dus à J.Pearl [Pearl(1982)] à l'université UCLA et une équipe de recherche danoise (Lauritzen, Spiegelhalter...) [Lauritzen et Spiegelhalter(1988)] à l'université d'Aalborg. L'objectif initial de ces travaux était d'intégrer la notion d'incertitude dans le raisonnement pour les systèmes experts. Les premières applications des réseaux bayésiens ne sont apparues qu'à partir des années 90 [Heckerman *et al.*(1992)].
Les réseaux bayésiens ont été appliqués dans plusieurs domaines :

- Santé : Les premières applications ont été développées dans le domaine du diagnostic médical. Dans le cadre du projet Human Genome du gouvernement américain, le National Health Institute et l'institut de technologie israélien Technion ont mis au point une méthode fondée sur l'utilisation des techniques d'inférence Bayésienne appliquée à la localisation des gènes, à partir de la localisation de gènes connus, et de l'analyse d'arbre généalogique [Becker *et al.*(1996)]

- Industrie : Dans le domaine industriel les réseaux bayésiens présentent certains avantages par rapport aux autres techniques d'intelligence artificielle. Par exemple la société danoise Hugin a développé pour le compte de Lockheed Martin le système de contrôle d'un véhicule sous-marin autonome. Ce système évalue en permanence les capacités du véhicule à réagir à certains types d'événements [Becker et Naim(1999)].

- Défense : La fusion des données est particulièrement un domaine d'application privilégié des réseaux bayésiens, grâce à leur capacité à prendre en compte des données incomplètes ou incertaines, et à guider la recherche ou la vérification de ces informations. La société Mitre a développé un système de défense

tactique embarqué pour les navires de guerre de la marine américaine. Ce système analyse les informations sur les missiles qui menacent le navire et décide des ripostes à adopter. Ce système permet en particulier de gérer les menaces multiples , qui peuvent générer des conflits sur l'affectation des armes [Becker et Naim(1999)].

Les modèles graphiques sont une union entre la théorie des probabilités et la théorie des graphes. Ce sont des graphes dont les nœuds représentent des variables aléatoires et les arcs représentent les dépendances conditionnelles. Par conséquent, ils fournissent une représentation compacte de la distribution de la probabilité jointe. Par exemple si on a N variables aléatoires binaires, la probabilité jointe $P(X_1, ..., X_N)$ a besoin de $O(2^N)$ paramètres, tandis que le modèle graphique peut en avoir besoin exponentiellement moins, selon les hypothèses de dépendance demandées. Ceci peut beaucoup aider dans les phases d'inférence et d'apprentissage. On distingue deux types de modèles graphiques :

- les modèles graphiques non dirigés (champs de Markov ou Markov Random Fields).
- les modèles graphiques dirigés (orientés) ou Réseaux Bayésiens.

Plus généralement on peut avoir plusieurs sortes de modèles graphiques, la figure Fig.3.1 montre la hiérarchie reliant les différentes sortes de modèles graphiques [Jordan(1998), Lauritzen *et al.*(1999), Jensen(2001)].
Dans ce chapitre nous allons définir les réseaux bayésiens et ensuite nous présenterons l'inférence et l'apprentissage.

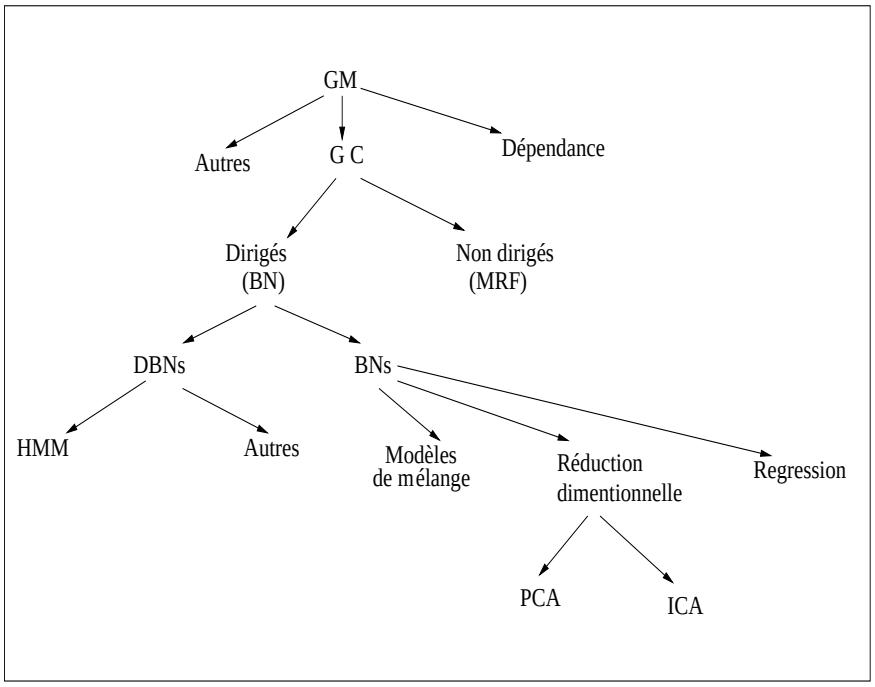


Fig. 3.1 – Hiérarchie partielle reliant les différents types de modèles graphiques. GM = Modèles graphiques, GC = graphes chaînés, BN = Réseaux Bayésiens, MRF = Champs de Markov, DBN = Réseaux bayésiens dynamiques, PCA = Analyse en composantes principales, ICA = Analyse en composantes indépendantes, Maxent = Maximum d'entropie

3.2 Représentation graphique de la causalité

Un réseau Bayésien (réseau probabiliste, **Bayesian Belief Network** ou **Bayesian Network**) est un modèle représentant des connaissances incertaines sur un phénomène complexe, et permettant, à partir des données, un véritable raisonnement. Un réseau Bayésien a pour objectif d'acquérir, de représenter et d'utiliser la connaissance. Il est constitué de deux composantes :

- un graphe causal, orienté, acyclique, dont les nœuds sont des variables d'intérêt du domaine, les arcs des relations de dépendance entre ces variables. L'ensemble des nœuds et des arcs forme ce que l'on appelle la structure du réseau Bayésien. C'est la représentation qualitative de la connaissance.
- un ensemble de distributions locales de probabilité qui sont les paramètres du réseau. Pour chaque nœud on dispose d'une table de probabilité $P(variable/parents(variable))$ qui représente la distribution locale de probabilité. Il faut remarquer que l'état de chaque nœud ne dépend que de l'état de ses parents. Il s'agit de la représentation quantitative de la connaissance.

On peut décrire un réseau Bayésien comme un système expert probabiliste. Dans un BN, un arc de A vers B peut être interprété par 'A cause B', les cycles ne sont pas autorisés, et le graphe est un graphe acyclique orienté (pour un exemple voir figure Fig.3.2). De plus un nœud est conditionnellement indépendant de ses non-descendants sachant ses parents.

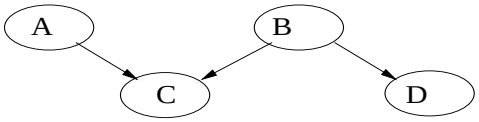
3.2.1 Exemple

Nous allons commencer par l'exemple suivant [Pearl(1988), Finn(1996)] :

> *Ce matin-là le temps est clair et sec, M.X sort de sa maison. Il s'aperçoit que la pelouse de son jardin est humide. Il se demande s'il a plu la nuit, ou s'il a simplement oublié de débrancher son arroseur automatique. Il jette un coup d'œil à la pelouse de son voisin, et s'aperçoit qu'elle est également humide. Il en déduit alors qu'il a plu, et il décide de partir au travail sans vérifier son arroseur automatique.*

La représentation graphique du modèle causal utilisé est dans la figure Fig3.2. Cette figure (3.2) représente un réseau Bayésien simple contenant quatre variables binaires, et on peut écrire aussi :

$$P(A, B, C, D) = P(A).P(B).P(C \mid A, B).P(D \mid B)$$



A Arroseur en marche
B Il a plu pendant la nuit
C Herbe du jardin humide
D Herbe du jardin voisin humide

P(A=V) = 0.4 P(A=F) = 0.6 P(B=V) = 0.7 P(B=F) = 0.3

A	B	P(C=F)	P(C=V)
V	V	0.3	0.7
V	F	0.05	0.95
F	V	0.4	0.6
F	F	0.78	0.22

B	P(D=V)	P(D=F)
F	0.5	0.5
V	0.58	0.42

FIG. 3.2 – Représentation graphique du modèle

3.2.2 Circulation de l'information

La représentation graphique la plus intuitive de l'influence d'un événement est de relier la cause à l'effet par une flèche orientée. S'il existe une relation causale de A vers B, toute information sur A peut modifier la connaissance sur B, et réciproquement, toute information sur B peut modifier la connaissance sur A. En présence d'un graphe plus complexe, il est essentiel de conserver à l'esprit que l'information ne circule pas seulement dans le sens des flèches.

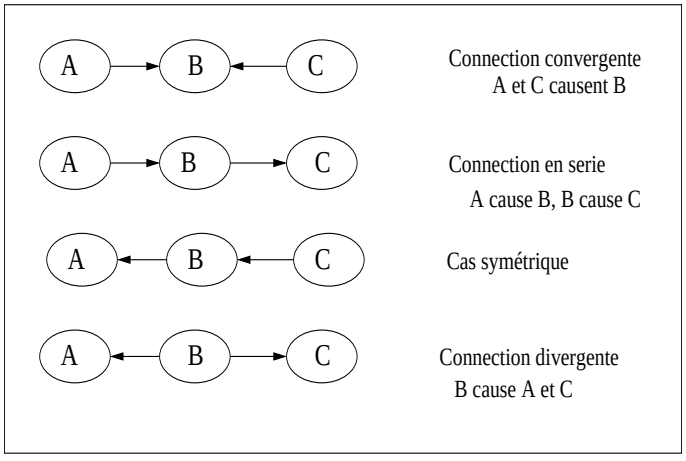


FIG. 3.3 – Modes de connexion

Dans un graphe on peut rencontrer plusieurs modes de connexion entre les nœuds comme le montre la figure Fig.3.3. Nous allons étudier comment l'information circule

au sein d'un graphe causal. Dans l'exemple précédent, on voit que l'information a circulé uniquement dans le sens effet $\longrightarrow$ cause, par exemple la connaissance de D (herbe du jardin humide) renforce la croyance de la cause B (il a plu). De plus cet exemple nous montre que l'information peut suivre des chemins à première vue contre-intuitifs lorsqu'elle se propage dans un réseau de causalités (voir figure Fig.3.4).

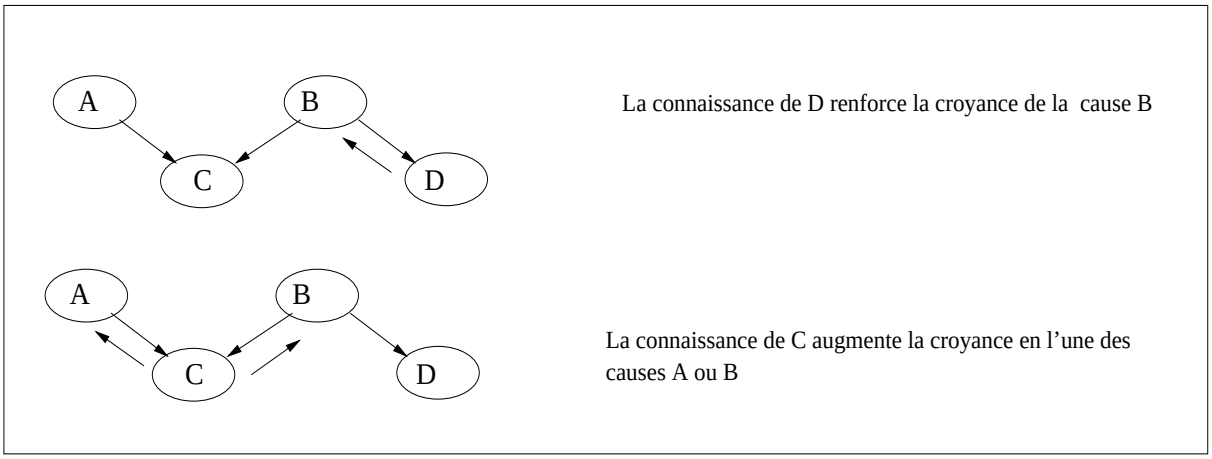


FIG. 3.4 – Circulation de l'information dans le graphe de la figure (3.3)

3.3 Définitions et propriétés

Nous présentons ici les définitions d'un réseau Bayésien, de l'indépendance conditionnelle et de la d-séparation. On commence par la définition d'un réseau Bayésien.

Définition 12 *:*
Soient

- *un graphe acyclique orienté G = (V,A), où V est l'ensemble des nœuds de G et A l'ensemble des arcs de G.*
- *un espace probabilisé fini (Ω, Z, P), n variables aléatoires* $(X_i)_{1\leq i\leq n}$.

(G, P) est un **réseau Bayésien** *si et seulement si*

- *il existe une bijection entre les nœuds du graphe G et les variables* (X_i).
- *la propriété suivante (appelée propriété de factorisation), est vérifiée*

$$P(X_1, X_2, X_3, ..., X_n) = \prod_{1\leq i\leq n} P(X_i|C(X_i)) \tag{3.1}$$

où $C(X_i)$ *est l'ensemble des causes (parents) de* X_i *dans le graphe G.*

3.3.1 Indépendance conditionnelle

Définition 13 *:*
Soient deux variables aléatoires X et Y. On dit que X et Y sont **indépendantes conditionnellement** *à Z et on note $X \perp Y \mid Z$, si l'une des propriétés équivalentes suivantes est vérifiée :*

$$P(X \mid Z, Y) = P(X \mid Z) \tag{3.2}$$
$$P(X, Y \mid Z) = P(X \mid Z) * P(Y \mid Z) \tag{3.3}$$

Propriété 1 *:*
Soient X,Y,Z,W des variables aléatoires, on a les propriétés suivantes :

$$\textit{Symétrie} \qquad X \perp Y \mid Z \iff Y \perp X \mid Z \tag{3.4}$$
$$\textit{Décomposition} \qquad X \perp Y \cup W \mid Z \Rightarrow X \perp Y \mid Z \tag{3.5}$$
$$\textit{Union faible} \qquad X \perp (Y \cup W) \mid Z \Rightarrow X \perp W \mid (Z \cup Y) \tag{3.6}$$
$$\textit{Contraction} \qquad X \perp Y \mid Z \wedge X \perp W \mid (Z \cup Y) \Rightarrow X \perp (Y \cup W) \mid Z \tag{3.7}$$

3.3.2 D-séparation

Définition 14 *:*
soient (X,Y,Z) des nœuds du graphe G =(V,A). On dit que X et Y sont d-séparés *par Z si pour tout chemin entre Xet Y, l'une au moins des deux conditions suivantes est vérifiée :*

- *Le chemin converge en un nœud W, tel que $W \neq Z$, W n'est pas une cause directe de Z.*
- *Le chemin passe par Z, est soit divergent, soit en série au nœud Z.*

X est d-séparé de Y par Z est noté par $< X|Z|Y >$.

Définition 15 *:*
soient (X,Y) deux nœuds du graphe G =(V,A), et soit $\mathbf{Z}$ *un ensemble de nœuds de ce graphe. Soit S un chemin entre les nœuds X et Y. On dit que S est* d-séparé *par* $\mathbf{Z}$ *si au moins l'une des deux conditions suivantes est vérifiée :*

- *Le chemin converge en un nœud W, tel que $W \notin \mathbf{Z}$ et $\forall Z \in \mathbf{Z}, W \notin C^+(Z)$ ou $C^+(Z)$ l'ensemble des ascendants de Z.*

– *Le chemin passe par un nœud $Z \in \mathbf{Z}$, et il est soit divergent, soit en série en ce nœud.*

Définition 16 *:*
soient (X,Y) des nœuds du graphe G =(V,A), et **Z** *un ensemble de nœuds de ce graphe. On dit que X et Y sont* d-séparés *par* **Z** *si tous les chemins entre X et Y sont d-séparés par* **Z**.

Définition 17 *:*
*Soient (***X, Y, Z***) trois ensembles des nœuds du graphe G = (V, A). On dit que* **X** *et* **Y** *sont* d-séparés *par* **Z**, *si tous les éléments de* **X** *sont d-séparés par* **Z** *de tous les éléments de* **Y**.

Remarque 4 *:*
Deux variables X, Y sont d-séparées s'il existe une variable Z telle que tout chemin entre X,Y passe par Z.

Dans le graphe de la figure Fig.3.5, A et D sont d-séparés par B car le chemin orienté allant de A vers D passe par B. Et de même le noeud D d-sépare les noeuds ABC des noeuds EFG.

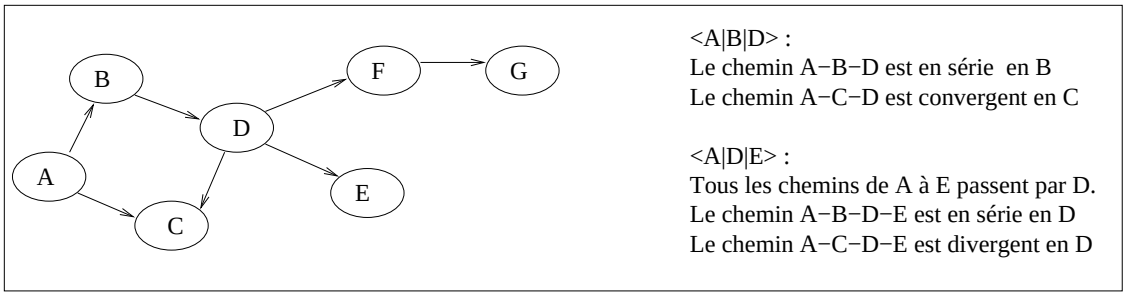


FIG. 3.5 – Exemple de D-séparation

Propriété 2 *:*
Soit G=(V,A) un graphe connecté acyclique orienté (CDAG). Soient N_1,N_2 et W trois sous-ensembles de V. Soient $W_1 \subset N_1$ et $W_2 \subset N_2$.
Si W d-sépare N_1 et N_2 alors W d-sépare W_1 et W_2

Propriété 3 *:*
Soient G=(V,A) un CDAG, u un nœud, V_1 et V_2 deux sous-ensembles de V fermés

dans G et disjoints tel que $u \notin V_1, u \notin V_2$.
Soit W un ensemble de nœuds vérifiant : $u \notin W$ *et* $W \cap F^-(u) = \varnothing$.
Soit U un ensemble de nœuds tel que $u \in U$.
On a les propriétés suivantes :

- *Si* $V_1 \cap C(u) \neq \varnothing$ *et* $V_2 \cap F(u) \neq \varnothing$ *alors U d-sépare* V_1 *et* V_2.
- *Si* $V_1 \cap F(u) \neq \varnothing$ *et* $V_2 \cap F(u) \neq \varnothing$ *alors U d-sépare* V_1 *et* V_2.
- *Si* $V_1 \cap C(u) \neq \varnothing$ *et* $V_2 \cap C(u) \neq \varnothing$ *alors W d-sépare* V_1 *et* V_2.

où C(u) est l'ensemble des parents de u, F(u) est l'ensemble des enfants de u et $F^-(u)$ *est l'ensemble des descendants de u.*

Définition 18 *:*
Soit une variable V, la couverture de Markov *de v (Markov blanket) est un ensemble qui se compose de V, ses parents, ses fils et les variables ayant un enfant commun avec V.*

L'indépendance conditionnelle et la D-séparation sont liées par le théorème suivant :

Théorème 1 *:*
soit B= (G,P), un réseau Bayésien. Soient $X \subset V$, $Y \subset V$, $Z \subset V$, *trois sous ensembles des nœuds de G. Si X et Y sont d-séparés dans G par Z, alors X et Y sont indépendants conditionnellement à Z i.e* $X \perp Y \mid Z$.

La démonstration est donnée dans [Becker et Naim(1999), Pearl(1988)].
Ce théorème est très important car il permet de limiter les calculs de probabilités grâce à certaines propriétés du graphe.

Remarque 5 *(Propriété locale de Markov [Russell et Norvig(2002), Murphy(2002)])*
Soit un nœud N d'un Réseau Bayésien on a :

1. *N est indépendant conditionnellement à tous les autres nœuds du réseau non appartenant à la couverture de Markov étant donné sa couverture de Markov.*
2. *N est indépendant conditionnellement aux nœuds non descendants de N étant donné les parents de N.*

Ces remarques sont représentées dans la figure Fig.3.6.

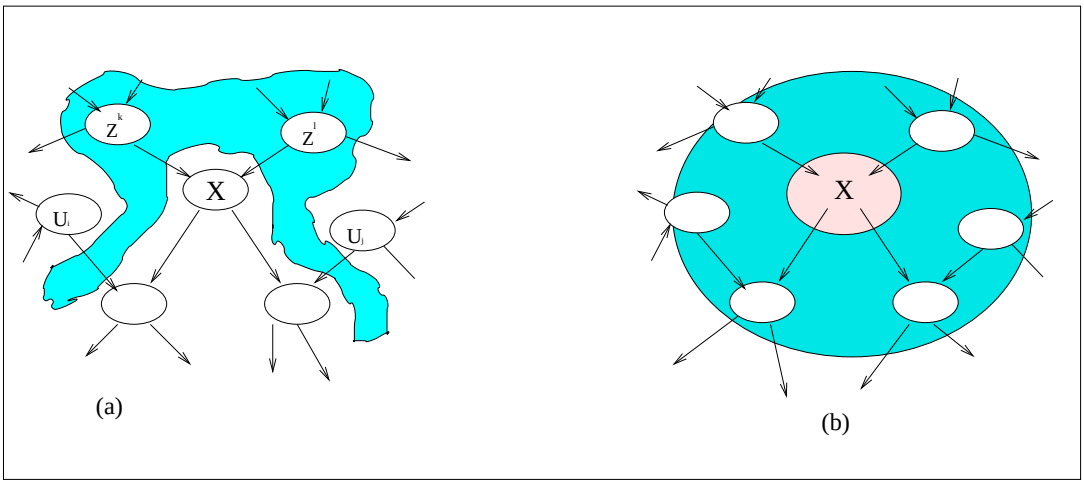


FIG. 3.6 – Propriétés locales de Markov. (a) Le nœud X est indépendant conditionnellement à ses non-descendants sachant ses parents (Z^k). (b) X est indépendant conditionnellement aux autres nœuds du réseau sachant la couverture de Markov [Murphy(2002)]

3.4 Inférence exacte dans un réseau Bayésien

L'utilisation essentielle des réseaux Bayésien est donc de calculer des probabilités conditionnelles d'événements reliés les uns aux autres par des relations de cause à effet. Cette utilisation s'appelle *inférence*. La correspondance qui existe entre la structure graphique et la structure probabiliste associée va permettre de ramener l'ensemble des problèmes de l'inférence à des problèmes de la théorie des graphes. Dans un réseau Bayésien, l'inférence consiste à propager une ou plusieurs informations au sein de ce réseau, pour en déduire comment sont modifiées les croyances concernant les autres nœuds. On l'utilise pour calculer des probabilités marginales sur les nœuds en absence ou présence de variables observées (évidence).
Supposons que nous disposons d'un réseau Bayésien défini par un graphe et la distribution de probabilité associée (G,P). Supposons que le graphe soit constitué de n nœuds notés $X = X_1, X_2, ..., X_n$. Le problème général de l'inférence consiste à calculer

$$P(X_i \mid Y) \quad \text{où} \quad Y \subset X, \quad X_i \notin Y$$

On voit bien que la complexité de ce problème dépend de la structure du réseau et de plus on va se limiter au cas de l'inférence dans un graphe orienté acyclique (DAG).
Les premiers algorithmes d'inférence ont été proposés dans [Pearl(1988)] concernant le cas d'arbre et polyarbre, ensuite généralisé par Jensen [Jensen *et al.*(1988)]

c'est le cas d'un arbre de jonction (JLO) (ce cas sera détaillé dans la suite). L'objectif de cette partie est donc de présenter l'algorithme JLO nommé aussi l'algorithme de la propagation de probabilité dans un arbre de regroupement (PPTC) [Lauritzen et Spiegelhalter(1988), Shenoy et Shafer(1990), Jensen *et al.*(1988)]. Cet algorithme est fondé sur les deux outils élémentaires que sont le théorème de Bayes et l'indépendance graphique. L'algorithme JLO convertit le réseau initial dans une seconde structure nommée **arbre de jonction** dont les nœuds sont des ensembles des variables de l'arbre initial (Cliques) et qui sont associées à des fonctions potentiels. Les probabilités recherchées sont calculées en manipulant cette nouvelle structure (utilisation des fonctions potentiels).
Vu la difficulté dans certains cas d'inférence, parfois on effectue des inférences approchées.

3.4.1 Élimination des variables

3.4.1.1 Généralité

Soit G = (V, A) un réseau Bayésien où $V = \{X_1, X_2, ..., X_n\}$ est l'ensemble des nœuds de G. En général le problème d'inférence consiste à calculer $P(X_W \mid X_Z)$ où X_W et X_Z sont des ensembles de variables. Cette expression ci peut être calculée à partir de la marginalisation de la probabilité jointe en utilisant la règle de Bayes :

$$\begin{aligned} P(X_W \mid X_Z) &= \frac{P(X_W, X_Z)}{p(X_Z)} \\ &= \frac{\sum_{r \notin W \cup Z} P(X_R = r, X_W, X_Z)}{\sum_{r \notin Z} P(X_R = r, X_Z)} \end{aligned} \tag{3.8}$$

De plus la probabilité jointe peut s'écrire sous la forme :

$$P(X_1, X_2, ..., X_n) = \prod_i P(X_i \mid C(X_i))$$

En remplaçant la probabilité jointe de l'équation 3.8 on trouve le $P(X_W \mid X_Z)$ cherché.

3.4.1.2 Exemple

Soit le réseau de la figure 3.7. Nous allons calculer par exemple le terme $P(G \mid B)$.

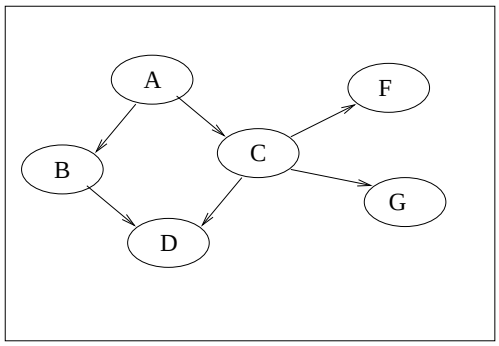


FIG. 3.7 – Réseau Bayésien

on procède de la manière suivante :

$$\begin{aligned} P(G \mid B) &= \frac{P(G,B)}{P(B)} \\ &= \frac{\sum_{A,C,D,F} P(A,B,C,D,F,G)}{\sum_{A,C,D,F,G} P(A,B,C,D,F,G)}) \end{aligned} \tag{3.9}$$

or

$$\begin{aligned} P(B) &= \sum_{A,C,D,F,G} P(A,B,C,D,F,G) \\ &= \sum_{A} P(A)P(B \mid A) \sum_{C} P(C \mid A) \sum_{D} P(D \mid B,C) \sum_{F} P(F \mid C) \sum_{G} P(G \mid C) \end{aligned} \tag{3.10}$$

on pose

$$\begin{aligned} \lambda_1(C) &= \sum_{F} P(F \mid C) \sum_{G} P(G \mid C) \\ \lambda_2(B,C) &= \sum_{D} P(D \mid B,C)\, \lambda_1(C) \\ \lambda_3(B,A) &= \sum_{C} P(C \mid A)\, \lambda_2(B,C) \end{aligned} \tag{3.11}$$

Ainsi

$$P(B) = \sum_{A} P(A)P(B \mid A)\lambda_3(B,A)$$

De la même façon on calcule $P(G,B)$.
Cette méthode s'appelle la méthode de *l'élimination* des variables.

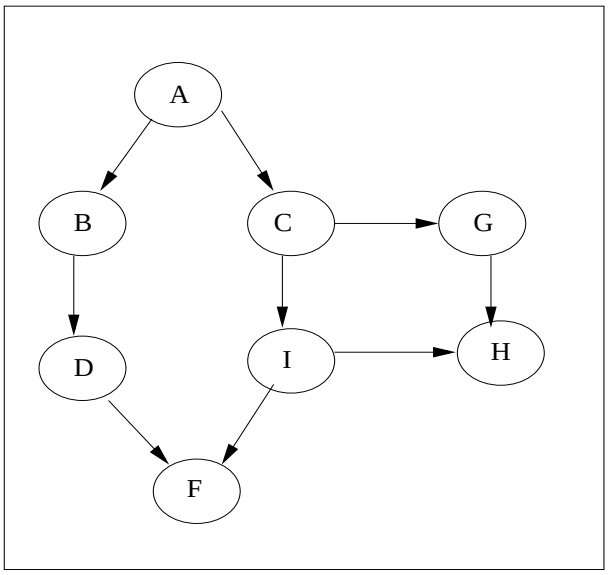
3.4.2 Définition et exemple

3.4.2.1 Définition

PPTC (**Probability Propagation in Trees of Clusters**) est une méthode d'inférence probabiliste dans un réseau Bayésien. En général **l'inférence probabiliste** dans un réseau est le processus qui consiste à calculer $P(V = v \mid \zeta = e)$ noté $P(V \mid e)$ où V est une variable du réseau et ζ un ensemble de variables observées appelées observation (évidence).
Nous considérons l'exemple représenté dans la figure Fig.3.8 dont les variables sont binaires, un exemple d'inférence probabiliste devrait calculer $P(A = 1 \mid C = 1, I = 0)$.
Le PPTC fonctionne en plusieurs étapes :

- Conversion du BN en l'arbre de jonction associé.
- Calculs des fonctions potentiel associées à l'arbre de jonction.
- Les probabilités cherchées sont calculées en opérant dans l'arbre de jonction.

Pour calculer les probabilités, on passe par un calcul intermédiaire basé sur le calcul des fonctions potentiel. Nous donnons ci-dessous leur définition.



$$\mathrm{P(A)} = \begin{array}{c|c} \mathrm{a} & \mathrm{P(a)} \\ \hline 1 & 0.5 \\ 0 & 0.5 \end{array} \qquad P(B \mid A) = \begin{array}{c|cc} & P(b \mid a) & \\ \mathrm{a} & 1 & 0 \\ \hline 1 & 0.5 & 0.5 \\ 0 & 0.4 & 0.6 \end{array}$$

$$P(C \mid A) = \begin{array}{c|cc} & P(c \mid a) & \\ \mathrm{a} & 1 & 0 \\ \hline 1 & 0.7 & 0.3 \\ 0 & 0.2 & 0.8 \end{array} \qquad P(D \mid B) = \begin{array}{c|cc} & P(d \mid b) & \\ \mathrm{b} & 1 & 0 \\ \hline 1 & 0.9 & 0.1 \\ 0 & 0.5 & 0.5 \end{array}$$

$$P(G \mid C) = \begin{array}{c|cc} & P(g \mid c) & \\ \mathrm{c} & 1 & 0 \\ \hline 1 & 0.8 & 0.2 \\ 0 & 0.1 & 0.9 \end{array} \qquad P(I \mid C) = \begin{array}{c|cc} & P(i \mid c) & \\ \mathrm{c} & 1 & 0 \\ \hline 1 & 0.3 & 0.7 \\ 0 & 0.6 & 0.4 \end{array}$$

$$P(F \mid D, I) = \begin{array}{cc|cc} & & P(f \mid d, i) & \\ \mathrm{d} & \mathrm{i} & 1 & 0 \\ \hline 1 & 1 & 0.01 & 0.99 \\ 1 & 0 & 0.01 & 0.99 \\ 0 & 1 & 0.01 & 0.99 \\ 0 & 0 & 0.99 & 0.01 \end{array} \qquad P(H \mid I, G) = \begin{array}{cc|cc} & & P(h \mid i, g) & \\ \mathrm{d} & \mathrm{i} & 1 & 0 \\ \hline 1 & 1 & 0.05 & 0.95 \\ 1 & 0 & 0.95 & 0.05 \\ 0 & 1 & 0.95 & 0.05 \\ 0 & 0 & 0.95 & 0.05 \end{array}$$

Fig. 3.8 – Réseau Bayésien BN

3.4.2.2 Potentiels

Soit un ensemble de variables U on définit la fonction potentiel par

$$\Phi_U : \begin{array}{ccc} U & \longrightarrow & \mathbb{R}^+ \\ u & \longrightarrow & \Phi_U(u) \end{array}$$

U est appelé *domaine* de la fonction Φ_U il est aussi noté dom(U).
On définit deux opérations sur les potentiels : la *marginalisation* et la *multiplication.*
Pour cela on suppose qu'on a deux ensembles de variables X et Y tels que $X \subset Y$ et son potentiel Φ_Y.
La *marginalisation* de Φ_Y dans X est un potentiel noté Φ_X tel que :

$$\Phi_X = \sum_{Y \setminus X} \Phi_Y$$

La *multiliplication* de deux potentiels Φ_1 et Φ_2 est un potentiel qui a les propriétés suivantes :

1. $\text{dom}(\Phi_1\ \Phi_2) = \text{dom}(\Phi_1) \cup \text{dom}(\Phi_2)$.
2. $\Phi_1\Phi_2 = \Phi_2\Phi_1$.
3. $(\Phi_1\Phi_2)\Phi_3 = \Phi_1(\Phi_2\Phi_3)$.
4. $\Phi_\varnothing = 1$ et $\mathbf{1}\ .\ \Phi = \Phi$.

Remarque 6 *:*
Les potentiels peuvent être considérés comme des matrices ou des tables de probabilités (conditionnelles) mais pas nécessairement des probabilités (conditionnelles).

Définition 19 *:*
Soient $G = (V, E)$ un graphe et $W \subset V$. W est une clique *si et seulement si*

$$\forall (u, v) \in W \times W, (u, v) \in E.$$

Propriété 4 *:*
Étant donné un BN (V,A) avec $V = (V_1, ..., V_n)$, un arbre non dirigé τ et deux cliques (ensemble non vide de variables du BN) X et Y dans τ. Toutes les cliques sur le chemin entre X et Y contiennent $X \cap Y$, et aussi pour chaque variable $V_i \in V$, la famille de V_i notée F_{V_i} (V_i et ses parents $C(V_i)$) est au moins incluse dans un groupement.

3.4.3 Structure secondaire d'un BN

Étant donné un BN (V,A) avec $V = (V_1, ..., V_n)$. On définit cette structure secondaire par deux composantes graphique et numérique :

1. La composante graphique nommée *Arbre de jonction* se compose de :
   - Un arbre non dirigé τ tel que chaque clique (nœud dans τ) doit satisfaire à la propriété 4 (Join tree property).
   - Des séparateurs : c'est un ensemble constitué de l'intersection de deux cliques adjacentes (**Sepsets**).
2. La composante numérique est décrite en utilisant les fonctions de potentiels associées aux cliques et aux séparateurs de l'arbre de jonction de la manière suivante :
   (a) Chaque clique X est associée à un potentiel noté Φ_X
   (b) Chaque séparateur S est associé à un potentiel noté Φ_S

   Ces fonctions de potentiels doivent satisfaire aux propriétés suivantes :
   - pour chaque clique X et son séparateur voisin S on a :

$$\sum_{X \setminus S} \Phi_X = \Phi_S \tag{3.12}$$

     Si l'équation (3.12) est vérifiée pour une clique et son séparateur voisin, on dit que Φ_S est **uniforme (ou consistant)** avec Φ_X. Quand cette uniformité se tient pour chaque paire clique-séparateur, on dit que l'arbre de jonction est *localement homogène*.
   - les potentiels codent la distribution jointe P(V) du réseau par

$$P(V) = \frac{\prod_i \Phi_{X_i}}{\prod_j \Phi_{S_j}} \tag{3.13}$$

     avec Φ_{X_i} et Φ_{S_j} sont les potentiels de clique et de séparateur respectivement.

De plus la seconde structure a une importante propriété :

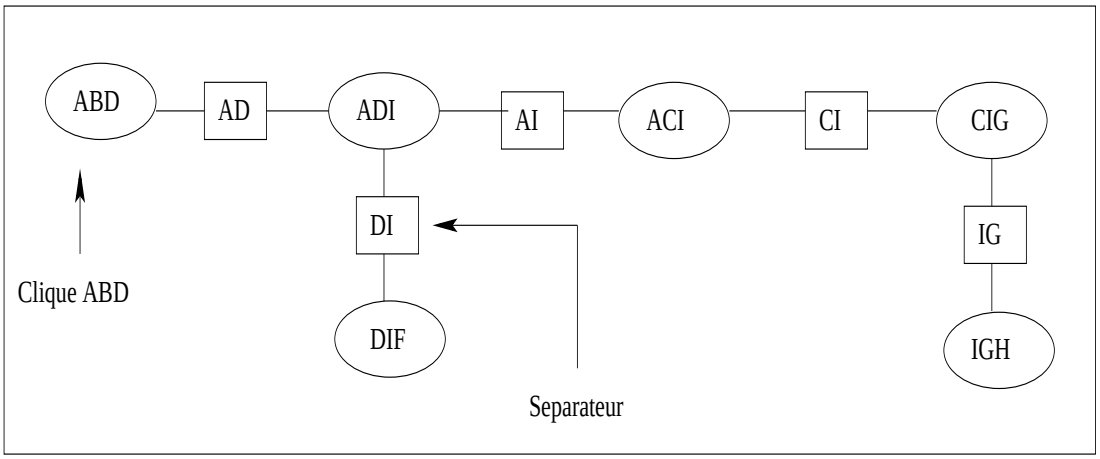
Propriété 5 *:*
Pour chaque clique ou séparateur X, on a $\phi_X = P(X)$.

En utilisant cette propriété on peut calculer les probabilités marginales pour chaque variable V_i du réseau :

$$P(V_i) = \sum_{X \setminus V_i} \phi_X \tag{3.14}$$

Exemple

La figure Fig.3.9 illustre l'arbre de jonction obtenu à partir du BN de la figure Fig.3.8. Cet arbre de jonction contient les cliques $\{ABD, ACE, ADE, CEG, DEF, EGH\}$



$\phi_{ABD} =$

a	b	d	$\phi_{ABD}(a,b,d)$
1	1	1	0.225
1	1	0	0.025
1	0	1	0.125
1	0	0	0.125
0	1	1	0.180
0	1	0	0.020
0	0	1	0.150
0	0	0	0.150

$\phi_{AD} =$

a	d	$\phi_{AD}(a,d)$
1	1	0.35
1	0	0.15
0	1	0.33
0	0	0.17

Fig. 3.9 – Exemple d'un arbre de jonction

et les séparateurs $\{AD, AE, CE, DE, EG\}$, chacun de ces ensembles (cliques, séparateurs) a un potentiel associé ϕ, de plus par exemple ϕ_{ABD} et ϕ_{AD} satisfont à la propriété de l'uniformité locale car

$$\phi_{AD} = \sum_{B} \phi_{ABD}$$

La distribution jointe est donnée par

$$P(V) = \frac{\phi_{ABD} \cdot \phi_{ACE} \cdot \phi_{ADE} \cdot \phi_{CEG} \cdot \phi_{DEF} \cdot \phi_{EGH}}{\phi_{AD} \cdot \phi_{AE} \cdot \phi_{CE} \cdot \phi_{DE} \cdot \phi_{EG}}$$

3.4.4 Construction de l'arbre de jonction

Dans cette partie nous allons commencer par un graphe dirigé acyclique (DAG) d'un BN et ensuite nous allons appliquer des transformations graphiques qui nous donnent la structure finale : arbre de jonction. Ces transformations impliquent un certain nombre de structures intermédiaires et peuvent être récapitulées par :

1. Moralisation : construction d'un graphe non dirigé, appelé *graphe moral.*
2. Triangulation : ajout sélectif des arcs au graphe moral pour former un graphe triangulé.
3. À partir du graphe triangulé, on construit des ensembles de nœuds appelés cliques. Chaque nœud contient une ou plusieurs variables du BN original.
4. Pour construire l'arbre de jonction, on connecte les cliques pour former un arbre non dirigé, qui vérifie la propriété 4 et une autre propriété supplémentaire.

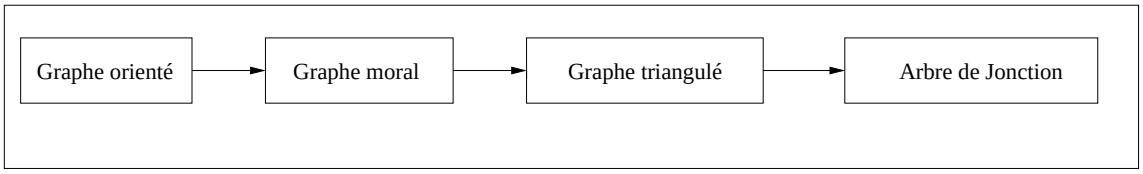


FIG. 3.10 – Transformations intermédiaires pour l'obtention de l'arbre de regroupement

La figure Fig.3.10 résume ces transformations intermédiaires.

Remarque 7 *:*
Les étapes 2 et 4 ne sont pas déterministes donc on peut construire plusieurs arbres de regroupement à partir du même DAG.

L'arbre de regroupement est un arbre dont les nœuds (cliques) sont des ensembles de nœuds du graphe original, vérifiant la propriété 4.

3.4.4.1 Moralisation

L'étape de la moralisation consiste à marier les parents de chaque nœud deux à deux, puis à éliminer les directions dans le graphe obtenu.

Définition 20 *:*
Soit G = (V, A) un graphe orienté.
On dit que le graphe M = (V, E_M) est le graphe moral de G si et seulement si :

- *M n'est pas orienté.*
- $A \subset E_M$.
- $\forall (u,v) \in V \times V, F(u) \cap F(v) \neq \varnothing \Longrightarrow (u,v) \in E_M$

Pour la partie algorithmique on a :
Soit **G** un graphe acyclique orienté d'un réseau Bayésien. Le graphe moral $\mathbf{G}_M$ qui correspond à **G** est construit [Lauritzen *et al.*(1990)] [Lauritzen et Spiegelhalter(1988)] de la façon suivante :

1. Création d'un graphe non orienté $\mathbf{G}_u$ en copiant **G** sans les directions des arcs.
2. Création de $\mathbf{G}_M$ à partir de $\mathbf{G}_u$: pour chaque nœud V et ses parents C(V) dans **G** nous connectons chaque paire de nœuds dans C(V) en ajoutant un arc non orienté à $\mathbf{G}_u$.

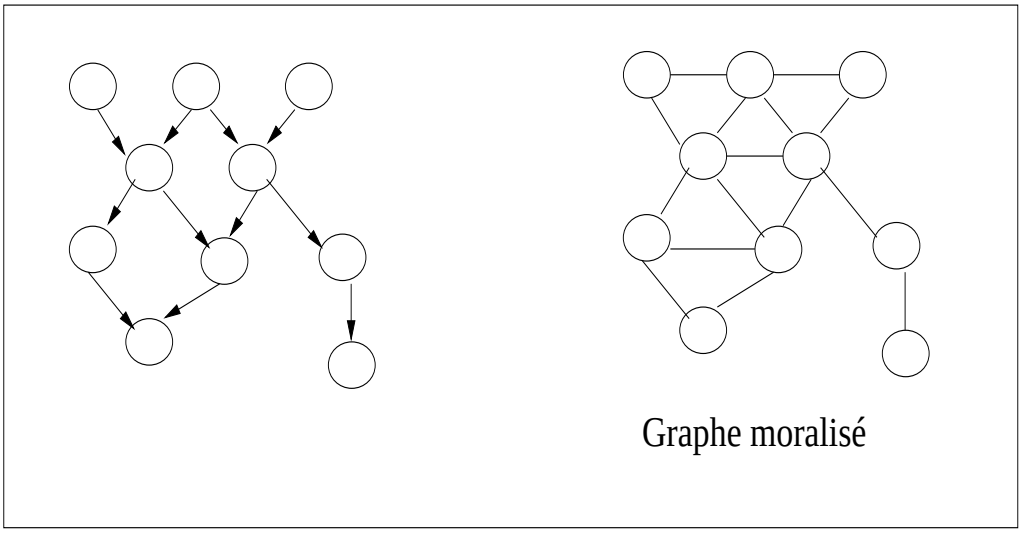


Fig. 3.11 – Construction d'un graphe moral

La figure Fig.3.11 donne un exemple de l'étape de la moralisation.

Remarque 8 :
Si un nœud a plus de deux parents il faut connecter ses nœuds (parents) deux à deux.

3.4.4.2 Triangularisation du graphe moral

Un graphe non orienté est triangulé si chaque cycle de longueur quatre ou plus, contient un arc qui relie deux nœuds non adjacents dans le cycle.

Définition 21 :
Soit G = (V, E) un graphe non orienté. Un graphe T = (V, E_T) est un graphe triangulé de G si et seulement si

- *T n'est pas orienté.*
- *$E \subset E_T$.*
- *pour tout cycle $[v_0, v_1, ..., v_n, v_0]$ de longueur supérieure ou égale à 4, il existe $i > j + 1$ tel que $(v_i, v_j) \in E_T$ est un arc.*

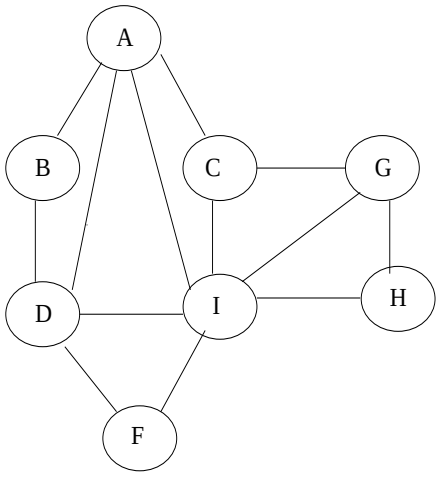
Remarque 9 :
Si G=(V,E) est triangulé alors pour tout graphe restreint G_W à $W \subset V$, est triangulé.

Nous allons décrire une procédure de triangularisation (algorithme d'élimination) élaborée par [Kjaerulf(1990)] :

1. Faire une copie de $\mathbf{G}_M$ qu'on appelle $\mathbf{G}'_M$.
2. Tant qu'il reste des nœuds dans $\mathbf{G}'_M$ on fait les étapes suivantes :
   -a sélectionner un nœud V de $\mathbf{G}'_M$
   -b ce nœud V et ses voisins dans $\mathbf{G}'_M$ forment une clique. On connecte tous les nœuds de cette clique. Pour chaque arc ajouté dans $\mathbf{G}'_M$, on ajoute le même dans $\mathbf{G}_M$
   -c enlever V de $\mathbf{G}'_M$.
3. $\mathbf{G}_M$, modifié par les arcs ajoutés dans les étapes précédentes, est triangulé.

La figure 3.12 montre les étapes de la triangularisation par l'algorithme précédent.

Remarque 10 :
Les ensembles d'élimination crées pendant la triangulation peuvent être rangés dans un arbre appelé arbre de bucket [Dechter(1998)] ou l'arbre d'élimination [Lauritzen et al.*(1999)]*



Graphe triangulé

Variable eliminée	Clique	Arc ajouté
H	IGH	aucun
G	CIG	aucun
F	DIF	aucun
C	ACI	(A,I)
B	ADB	(A, D)
D	ADI	aucun
I	AI	aucun
A	A	aucun

Algorithme d'élimination

FIG. 3.12 – Triangularisation du graphe moral

3.4.4.3 Cliques

Définition 22 *:*
Soient $G = (V, E)$ un graphe et W une clique.
W est une clique maximale si et seulement s'il n'existe aucun sur-ensemble $U \supset W$, tel que U soit une clique.

Ainsi on peut dire qu'une clique dans un graphe non orienté **G** est complète et maximale si elle est un sous-graphe *complet* et *maximal* tel que

Complet signifie que chaque paire de nœuds (variables) distincts est connectée par un arc.

Maximal signifie que la clique n'est pas complètement contenue dans un sous-graphe complet.

Remarque 11 *:*
Tous les nœuds d'une clique sont reliés deux à deux dans le graphe triangulé. Golumbic [Golumbic(1980)] a donné un algorithme efficace pour l'identification des cliques d'un graphe triangulé.

Dans la figure Fig.3.12 les cliques du graphe triangulé sont
$(EGH), (CEG), (DEF), (ACE), (ABD)$ *et* (ADE).

3.4.4.4 Arbre de jonction

Définition 23 *:*
Soit G = (v,E) un graphe orienté acyclique. Soient M = (V,E_M) le graphe moral associé à G, T =(V, E_T) le graphe triangulé associé à M. On dit que J= (V, A_i) est un arbre de jonction associé à G si et seulement si :

- *J est un arbre de regroupement sur V.*
- *Toute clique maximale dans T est un nœud de J.*

À présent nous n'avons plus besoin de graphe non orienté. Nous cherchons à construire un arbre optimal de jonction en connectant les cliques obtenues dans le paragraphe précédent. Nous rappelons que les séparateurs contiennent les variables communes à deux cliques connectées dans l'arbre de jonction.

Définition 24 *:*
Soient G un ensemble de cliques à partir d'un graphe non orienté et que ces cliques de G sont rangées dans un arbre T.
T est un arbre de regroupement si pour chaque paire de nœuds (u,v) de T , tous les nœuds dans le chemin entre v et u contiennent l'intersection $v \cap u$.

Définition 25 *:*
Soit G = (v,E) un graphe orienté acyclique. Soient M = (V,E_M) le graphe moral associé à G, T =(V, E_T) le graphe triangulé associé à M. On dit que J= (V, A_i) est un arbre de jonction *associé à G si et seulement si :*

- *J est un arbre de regroupement sur V.*
- *Toute clique maximale dans T est un nœud de J.*

Théorème 2 *:*
Soit T= (V,E) un graphe triangulé,alors il existe un arbre de jonction J =(V, A) sur T.

La figure 3.13 montre un exemple de l'arbre de regroupement on voit que (a) est un arbre de regroupement par contre (b) ne l'est pas car l'intersection des deux cliques (BCDE) et (CHGJ) est C qui n'appartient pas à la clique (DEFI).

Théorème 3 *:*
Un graphe non orienté G est triangulé si et seulement si les cliques de G peuvent être rangées dans un arbre de regroupement.

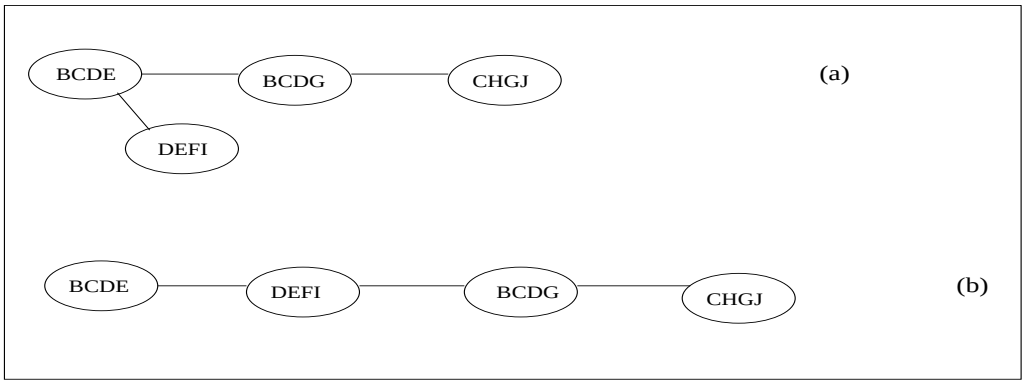


FIG. 3.13 – (a) : arbre de regroupement (b) : n'est pas un arbre de regroupement

En ce qui concerne la partie algorithmique on procède de la façon suivante :
Pour construire un arbre optimal de regroupement, nous devons connecter les cliques obtenues de sorte que l'arbre résultant des cliques satisfait à la propriété de l'arbre joint (Join tree property) et un critère que nous décrivons ci-dessous
[Jensen et Jensen(1994)] .
Étant donné un ensemble de n cliques, on peut construire un arbre de cliques en connectant itérativement chaque paire de cliques par un arc jusqu'à ce que les cliques soient toutes connectées par $n-1$ arcs.

1. construction de l'arbre des cliques
   (a) on commence avec un ensemble de n arbres, chaque arbre se compose d'une simple clique et d'un ensemble de séparateurs vide **S**.
   (b) pour chaque paire distincte de cliques **X** et **Y** :
   - nous créons un séparateur $S_{XY} = X \cap Y$
   - on insère $S_{XY} = X \cap Y$ dans **S**
   (c) on répète l'instruction (b) jusqu'à obtenir $n-1$ séparateurs tels que
   -a on sélectionne un séparateur S_{XY} de **S** suivant le critère indiqué ci-dessous . Puis on élimine S_{XY} de **S**.
   -b on insère S_{XY} entre **X** et **Y** uniquement si **X** et **Y** sont dans des arbres différents .
2. choix des séparateurs appropriés : on décrit comment choisir le futur séparateur en se basant sur les deux notions de **masse** et de **coût**
   – la masse d'un séparateur S_{XY} est le nombre des variables de $X \cap Y$
   – le coût d'un séparateur S_{XY} est la somme des poids de **X** et **Y** où le poids est défini par

(a) le poids d'une variable V est le nombre de ses valeurs d'états possibles

(b) le poids d'un ensemble de variables X est le produit des poids des variables de l'ensemble X

On peut maintenant sélectionner le futur séparateur de l'ensemble **S** , quand on exécute l'étape (c) :

1. l'arbre de clique résultant doit satisfaire à la propriété 4 et aussi on doit choisir chaque séparateur ayant la plus grande masse
2. quand deux séparateurs ou plus ont la même masse, on choisit le futur séparateur ayant le plus petit coût.

3.4.5 Principe de l'inférence sans observations

Définition 26 *:*
soit Φ *l'ensemble de potentiels associés au domaine du graphe triangulé G. L'arbre de jonction* **J** *est un arbre de regroupement du graphe G tel que :*

- chaque potentiel ϕ de Φ est associé à une clique contenant le dom(ϕ)
- chaque arc du graphe G a un séparateur approprié

Après avoir construit l'arbre de jonction, nous allons donner dans cette partie la procédure pour calculer sa composante numérique (c-a-d. les fonctions potentiels) pour former l'arbre de jonction. Sur cet arbre de jonction, on peut faire de l'inférence probabiliste avec ou sans observation (évidence). L'inférence consiste à calculer la distribution de probabilité $P(V)$ ou $P(V \mid e)$ dans le contexte de l'ensemble des observations **e**.

La figure Fig.3.14 illustre les principes de l'inférence sans observation.
Le fonctionnement se résume dans les étapes suivantes :

1. Transformation graphique (structure) : c'est la transformation qui donne l'arbre de regroupement.
2. Initialisation : associer les fonctions des potentiels à l'arbre de regroupement de telle sorte qu'il satisfasse à l'équation 3.13. Le résultat est un arbre de jonction modifié.
3. Propagation : exécuter une série de manipulations locales appelées **passage de messages** dans l'arbre de jonction. Ainsi le résultat de la propagation est l'arbre de jonction complet (mis à jour) qui satisfait aux équations 3.12 et 3.13.
4. Marginalisation : à partir de l'arbre de jonction complet on calcule P(V) pour chaque variable V.

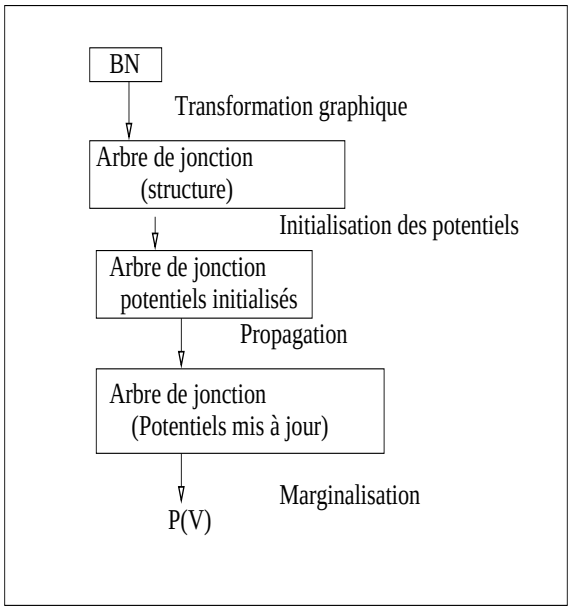


FIG. 3.14 – Principe de l'inférence

3.4.6 Initialisation

Le procédé suivant affecte le potentiel initial de l'arbre de jonction en utilisant les probabilités conditionnelles du réseau bayésien $P(V_i \mid C(V_i))$:

1. Pour chaque clique et séparateur $\mathbf{X}$ on place $\phi_X(X)$ à 1

$$\phi_X(X) \leftarrow 1$$

2. On procède pour chaque variable V_i de la façon suivante : on affecte V_i à une clique X qui contient F_{V_i} appelée clique parent, on multiplie $\phi_X(X)$ par $P(V_i \mid C(V_i))$:

$$\phi_X(X) \leftarrow \phi_X(X) P(V_i \mid C(V_i))$$

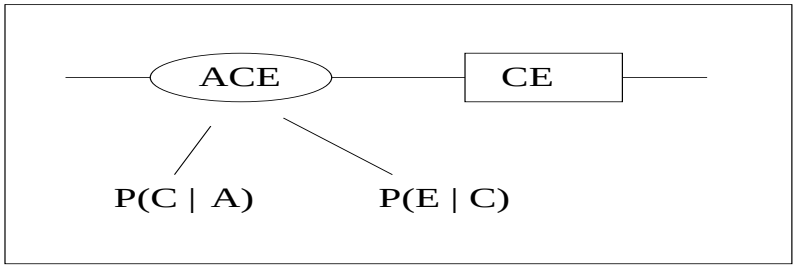
La procédure d'initialisation satisfait à l'équation 3.13 :

$$\frac{\prod_{i=1}^{N} \phi_{X_i}}{\prod_{j=1}^{N-1} \phi_{S_j}} = \frac{\prod_{l=1}^{M} P(V_l \mid C(V_l))}{1} = P(V)$$

où V est l'ensemble des variables du réseau, N est le nombre de cliques, M est le nombre des variables et ϕ_{X_i}, ϕ_{S_j} sont les potentiels de clique et de séparateur respectivement.

La figure 3.15 illustre la procédure de l'initialisation pour la clique ACE. Ainsi après l'initialisation on a :

$$\phi_{ACE} = P(C \mid A) P(E \mid C) \quad et \quad \phi_{CE} = 1$$



a	c	e	ϕ_{ACE} Initialisation
1	1	1	1 * 0.7 * 0.3 = 0.21
1	1	0	1 * 0.7 * 0.7 = 0.49
1	0	1	1 * 0.3 * 0.6 = 0.18
1	0	0	1 * 0.3 * 0.4 = 0.12
0	1	1	1 * 0.2 * 0.3 = 0.06
0	1	0	1 * 0.2 * 0.7 = 0.14
0	0	1	1 * 0.8 * 0.6 = 0.48
0	0	0	1 * 0.8 * 0.4 = 0.32

c	e	ϕ_{CE} Initialisation
1	1	1
1	0	1
0	1	1
0	0	1

Fig. 3.15 – Initialisation de la clique ACE et du séparateur CE de l'arbre de regroupement de la fig 3.12

3.4.7 Propagation

Après avoir initialisé les potentiels de l'arbre de jonction, nous allons exécuter la propagation afin d'avoir une mise à jour consistante de ces potentiels. La propagation se compose de plusieurs manipulations appelées **passage de messages** qui se produisent entre une clique **X** et sa clique voisine **Y**. Le passage de message de **X** à **Y** force le potentiel de croyance du séparateur S_{XY} à être consistant avec ϕ_X (3.12), tout en préservant l'invariance de l'équation 3.13. La propagation fait passer de chaque clique un message à chacun de ses voisins ; ces passages de message sont commandés de sorte que chaque passage de message préserve la mise à jour présentée par les passages de messages précédents. Quand on termine la propagation, chaque paire de clique-séparateur est mise à jour et l'arbre de jonction est complet.

3.4.7.1 Passage de message simple

On considère deux cliques adjacentes **X** et **Y** avec un séparateur **S** et leurs potentiels associés ϕ_X, ϕ_Y et ϕ_S. Le passage de message de **X** à **Y** se produit en deux étapes nommées projection et absorption :

1. **Projection** on affecte une nouvelle table à **S** , et on sauvegarde l'ancienne table :

$$\phi_S^{old} \leftarrow \phi_S$$

$$\phi_S \leftarrow \sum_{X \setminus S} \phi_X \tag{3.15}$$

2. **Absorption** on affecte une nouvelle table à **Y** en utilisant l'ancienne et la nouvelle table de **S** :

$$\phi_Y \leftarrow \phi_Y \frac{\phi_S}{\phi_S^{old}} \tag{3.16}$$

Remarque 12 *Pour une réalisation $S = s$, Jensen [Jensen(1996)] montre que*

$$\phi_S^{old}(s) = 0 \quad \textit{uniquement si} \quad \phi_S(s) = 0$$

Les équations 3.15 et 3.16 affectent des nouveaux potentiels à **S** et **Y**, cependant le terme à droite de l'équation 3.16 reste constant. La probabilité jointe du réseau s'écrit

$$\left(\frac{\prod_i \phi_{x_i}}{\prod_j \phi_{S_j}}\right) \frac{\phi_S^{old}}{\phi_S} \frac{\phi_Y}{\phi_Y^{old}} = \left(\frac{\prod_i \phi_{x_i}}{\prod_j \phi_{S_j}}\right) \frac{\phi_S^{old}}{\phi_S} \frac{\phi_Y^{old} \frac{\phi_S}{\phi_S^{old}}}{\phi_Y^{old}} = P(V)$$

3.4.7.2 Passage de message multiple

Étant donné un arbre de regroupement avec n cliques, l'algorithme PPTC de propagation commence par le choix arbitraire d'une clique ϕ_X et on exécute alors $2(n-1)$ passages de messages divisés en deux phases :

Rassemblement de l'observation (**Collect-Evidence**) dans cette phase chaque clique fait passer un message à ses cliques voisines dans la direction convergente vers **X**, en commençant avec les cliques les plus éloignées de **X** [Huang et Darwiche(1996)].

Distribution de l'observation (**Distribute-Evidence**) dans cette phase chaque clique fait passer des messages à ses cliques voisines dans la direction divergente de **X**, en commençant par la clique **X** [Huang et Darwiche(1996)].

Chacune de ces deux phases provoque $n-1$ passages de messages. Le fonctionnement du passage de message est que chaque clique fait passer des informations, codées dans les potentiels, à toutes les cliques du réseau.

Remarque 13 *Les cliques font passer des messages à leurs cliques voisines uniquement après réception de leurs messages. Cette condition assure la mise à jour (consistance) locale de l'arbre de jonction quand on termine la propagation [Jensen(1996)].*

3.4.7.3 Exemple

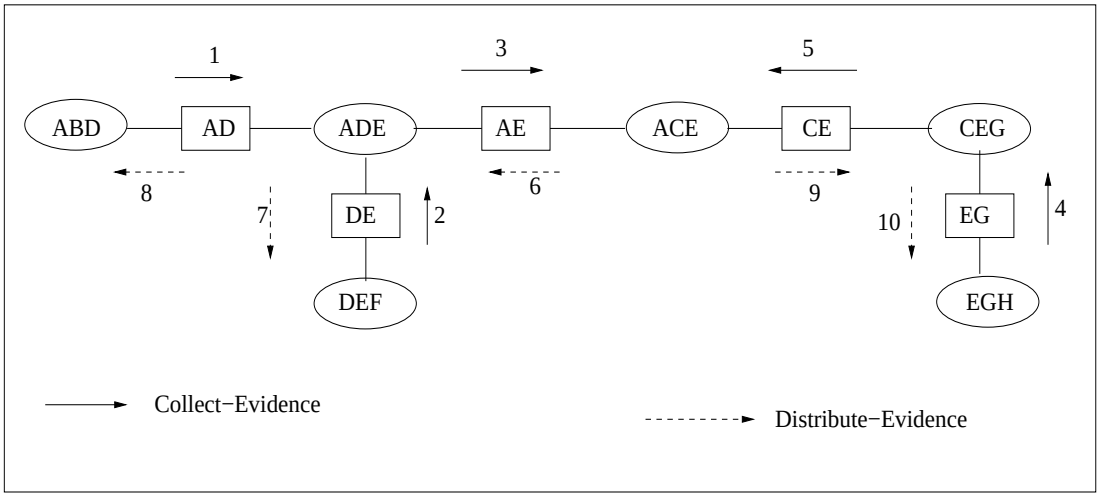


FIG. 3.16 – Ordre du passage de messages pendant la propagation à partir de la clique (ACE)

$\phi_{ABD} =$

a	b	d	$\phi_{ABD}(abd)$
1	1	1	0.225
1	1	0	0.025
1	0	1	0.125
1	0	0	0.125
0	1	1	0.180
0	1	0	0.020
0	0	1	0.150
0	0	0	0.150

$P(A) = \sum_{BD} \phi_{ABD} =$

a	P(a)
1	0.225 + 0.025 + 0.125 + 0.125 = 0.5
0	0.180 + 0.020 + 0.150 + 0.150 = 0.5

$P(D) = \sum_{AB} \phi_{ABD} =$

d	P(d)
1	0.225 + 0.125 + 0.180 + 0.150 = 0.680
0	0.025 + 0.125 + 0.020 + 0.150 = 0.320

FIG. 3.17 – Exemple de marginalisation

La figure Fig.3.16 illustre les étapes de la propagation dans l'arbre de jonction (3.8). Ici la clique ACE est la clique de commencement. Pendant la phase de Collect-Evidence les messages passent dans la direction convergente vers ACE, en commençant avec les cliques ABD, DEF et EGH. Pendant la phase Distribute-Evidence les messages passent dans la direction divergente de ACE, à partir de la clique ACE.

3.4.8 Marginalisation

Après avoir actualisé tous les potentiels (on obtient un arbre de jonction complet), on peut calculer P(V) pour chaque variable dans le réseau de la façon suivante :

1. Identifier la clique $\mathbf{X}$ contenant la variable V.
2. Calculer P(V) en marginalisant ϕ_X

$$P(V) = \sum_{X \setminus V} \phi_X.$$

La figure 3.17 est un exemple de la marginalisation.

3.5 Inférence avec observation

Dans cette partie nous allons voir comment calculer $P(V \mid e)$ dans le contexte de l'observation **e** (évidence).

Définition 27 *:*
Une observation est une déclaration de la forme $V = v$. On note une collection des observations par $\mathbf{E} = \mathbf{e}$, avec e est la réalisation de l'ensemble E de variables observées.

Définition 28 *:*
Soient V une clique, Φ_V son potentiel associé et S un séparateur voisin. Chaque séparateur S d'un arbre de regroupement fait passer deux messages dans les deux directions (convergente ou divergente) notés ψ_S et ψ^S. Soient $S_1, ..., S_l$ les autres séparateurs voisins de V. On suppose que chaque S_i reçoit un message ψ_i de V. Ainsi V peut passer le message $\sum_{V \setminus S} \Phi_V \psi_1 ... \psi_k$ à S et on dit que la direction V-S est activée.

La méthode de propagation consiste à répéter l'opération de passage de message à travers les directions activées (triggered).

Proposition 1 *:*
Si on répète le passage de message à travers les directions activées on ne doit pas arrêter le processus avant que les messages passent dans toutes les directions pour chaque arc. Dans ce cas on dit que l'arbre de jonction est complet.

Après avoir terminé la partie de passage de message on peut calculer la probabilité jointe de chaque clique dans le contexte de l'observation **e**.

Théorème 4 *:*
*Soient T un arbre de jonction représentant un BN sur l'univers U et **e** l'observation. On suppose que T est complet. Soient V une clique, Φ_V son potentiel associé, $S_1, ..., S_k$ ses séparateurs voisins et $\psi_1, ..., \psi_k$ les messages dirigés sur V , alors*

$$P(V, \mathbf{e}) = \prod \Phi_V \prod \psi_1 ... \psi_k. \tag{3.17}$$

Soit S un séparateur avec les ψ_S, ψ^S les messages de passage pour S. on a

$$P(S, \mathbf{e}) = \prod \psi_S \prod \psi^S. \tag{3.18}$$

Définition 29 *:*
Soit une variable V, la vraisemblance potentiel (Finding) de V, notée Λ_V est un potentiel sur $\{V\}$ ie.

$$\Lambda_V : \quad \begin{array}{rl} \{V\} \longrightarrow & 0,1 \\ v \longrightarrow & \Lambda_V(v) \end{array}$$

On peut coder un ensemble arbitraire d' observations (constituant l'évidence) **e** *en utilisant Λ_V pour chaque variable V de la façon suivante :*

1. *Si $V \in E$, et si V est observable on a :*

$$\Lambda_V(v) = \begin{cases} 1 & \text{si } v \text{ est une valeur observée de } V \\ 0 & \text{autres} \end{cases}$$

2. *Si $V \notin E$, et si V n'est pas observée alors*

$$\Lambda_V(v) = 1 \quad \forall v$$

Remarque 14 *:*
Λ_V est un tableau binaire où les valeurs non observées sont à 0. La table 3.1 montre comment les vraisemblances sont utilisées pour coder les observations $C = 1$ et $I = 0$ où C et I sont des variables du réseau de la figure Fig.3.8.

Variable		$\Lambda_V(v)$
V	v=1	v=0
A	1	1
B	1	1
C	1	0
D	1	1
I	0	1
F	1	1
G	1	1
H	1	1

TAB. 3.1 – Vraisemblance pour C = 1, I = 0

Dans ce paragraphe nous allons montrer le déroulement de l'inférence avec observation qui se base sur les étapes suivantes :

1. **Initialisation** on modifie l'initialisation de la partie précédente (initialisation de l'inférence sans observation) en introduisant une étape supplémentaire : pour chaque variable V, on initialise la vraisemblance de Λ_V.
2. **Entrée de l'observation** on code l'observation dans l'arbre de jonction, cette étape est conséquence de la modification des potentiels de l'arbre de jonction
3. **Normalisation** pour calculer $P(V \setminus e)$ pour une variable donnée V, on applique la marginalisation et une étape supplémentaire appelée *normalisation*

3.5.1 Initialisation

L'initialisation se compose des étapes suivantes ;

1. Pour chaque clique et séparateur **X**, on met ϕ_X à 1

$$\phi_X \leftarrow 1$$

2. Pour chaque variable V :

   on affecte V à une clique **X** qui contient F_V ; on multiplie ϕ_X par $P(V \setminus C(V))$

$$\phi_X \leftarrow \phi_X . P(V \mid C(V))$$

   on met chaque vraisemblance $\Lambda_V(v)$ à 1

$$\Lambda_V(v) \leftarrow 1$$

3.5.2 Entrée de l'observation

Dans ce paragraphe nous allons faire rentrer les observations dans l'arbre de jonction de la façon suivante :

1. Coder chaque observation $V = v$ comme une vraisemblance Λ_V^{new}
2. Identifier la clique contenant V (F_V)
3. Mise à jour des ϕ_X et Λ_V^{new} :

$$\begin{aligned} \phi_X &\leftarrow \phi_X \Lambda_V^{new} \\ \Lambda_V &\leftarrow \Lambda_V^{new} \end{aligned}$$

En entrant l'ensemble des observations **e** par la procédure décrite ci-dessus, on modifie les potentiels de l'arbre de jonction. Ainsi toutes les probabilités provenant de l'arbre de jonction sont des probabilités jointes des événements et l'observation **e**. Cela signifie qu'au lieu de calculer par exemple $P(X)$ et $P(V)$, on calcule $P(X, e)$ et $P(X, e)$.

Remarque 15 *L'arbre de jonction code maintenant $P(U, e)$ au lieu de $P(U)$.*

3.5.3 Marginalisation et normalisation

Après avoir propagé les informations dans l'arbre de jonction, on passe à la dernière étape. Pour chaque clique (ou séparateur) **X**, on a $\phi_X = P(X, e)$ où e est l'observation. Quand on marginalise le potentiel de clique ϕ_X d'une variable V, on obtient :

$$P(V, e) = \sum_{X \setminus V} \phi_X$$

Notre objectif est de calculer $P(V \mid e)$ probabilité de V étant donnée **e**. On obtient $P(V \mid e)$ à partir de $P(V, e)$ en normalisant $P(V, e)$:

$$P(V \mid e) = \frac{P(V, e)}{P(e)} = \frac{P(V, e)}{\sum_V P(V, e)} \tag{3.19}$$

la probabilité de l'observation $P(e)$ est la constante de normalisation.

3.5.4 Manipulation des observations

Pour calculer $P(V \mid e_2)$ avec e_2 est un ensemble d'observations différent de e_1, on recommence la procédure suivante : ré-initialisation des potentiels, entrée de l'observation e_2, propagation, marginalisation et normalisation. Cependant ce travail n'est pas nécessaire, car on peut directement modifier les potentiels de l'arbre de jonction obtenu sans observation pour répondre au changement de l'ensemble des observations. On peut imaginer un système dynamique dont l'arbre de jonction (après les modifications des potentiels) est l'état d'équilibre et tel que l'observation rentrante agit sur cet état d'équilibre.

Remarque 16 *:*
L'inférence peut être interprétée comme la propagation de certaines observations dans le réseau.
La complexité de l'inférence peut conduire à des temps de calculs prohibitifs pour des réseaux complexes. Il est impossible de calculer directement la loi de probabilité

d'un nœud ou d'effectuer une inférence plus complexe, d'où l'utilité d'introduire un nouveau type d'inférence nommé inférence approximative. *Ces méthodes d'approximation cherchent à estimer la distribution de probabilité complète représentée par le réseau, en effectuant des tirages aléatoires avec des lois simples [Jordan* et al.*(1999)], [Mackay(1999)] et [Jaakkola et Jordan(1999)]. Les deux grandes classes d'algorithme d'inférence approximative sont l'algorithme de* Monte Carlo *[Mackay(1999)] et l'algorithme* variationnel *[Jordan et Weiss(2001)].*

3.6 Apprentissage

3.6.1 Introduction

Les réseaux bayésiens sont un mode de représentation des connaissances fondé sur une description des relations entre les variables d'un domaine donné. Ce mode de représentation sous forme de distribution de probabilité permet l'utilisation la plus rationnelle possible de ces connaissances dans la plupart des situations. Si on appelle connaissance les relations entre les variables qui sont valables quelle que soit la situation, et informations les faits décrivant une situation donnée, donc l'inférence est l'outil qui permet de passer d'un modèle de connaissance et d'une situation à une conclusion. Dans ce paragraphe nous allons nous intéresser à la construction d'un modèle de connaissance, en vue de son utilisation future comme support d'inférence. Nous allons étudier les méthodes permettant la construction d'un tel modèle. Le problème confronté est alors celui de l'estimation de modèle à partir de données empiriques.

3.6.2 Notation et définition

Définition 30 *:*
La structure d'un réseau Bayésien est l'ensemble des arcs du graphe orienté sous-jacent au réseau.
Les paramètres d'un réseau Bayésien sont constitués par l'ensemble des probabilités conditionnelles qu'un nœud donné soit dans un état, étant donné les états de l'ensemble de ses parents.

Définition 31 *:*
Soit B=(G,P) un BN défini sur un graphe G=(V,A) et sur une distribution de probabilité associée à un produit de variables aléatoires finies $(X_i)_{i=1,n}$*. Un exemple associé au BN est un n-uplet* $(x_1^{k_1}, .., x_n^{k_n})$*. Une base d'exemples D est un ensemble ordonné d'exemples* $D = (E^k)_{k=1,N}$

Définition 32 *:*

Soit D une base de données associée au réseau B.
On note $N_{i,j,k}$ le nombre d'exemples $E^u = (x_1^{k_1(u)}, x_2^{k_2(u)}, .., x_n^{k_n(u)})$ dans D pour lesquels la variable X_i se trouve dans l'état k, alors que ses parents se trouvent dans l'état j, c-a-d le nombre d'exemples E_u tels que $j_i(u) = j$ et $k_i(u) = k$.
On note $N_{i,j}$ le nombre d'exemples de la base de données pour lesquels les parents de la variable X_i sont dans l'état j. D est complète par rapport à B si et seulement si $\forall i, j N_{i,j} \neq 0$

Notation

S	Structure du réseau Bayésien B
Θ	Ensemble des paramètres du réseau B
n	Nombre des nœuds, ou de variables du réseau B
X_i	Variable aléatoire discrète associée à B
$\mathbf{X}$	Vecteur des variables aléatoires X_i
$\mathbf{x}$	Réalisation de ce vecteur $\mathbf{x} = (x_1^{j_1}, .., x_n^{j_n})$
$\mathbf{N}$	Table de comptage
$\mathbf{N(x)}$	Nombre de cas dans la base D d'exemples où $\mathbf{X} = \mathbf{x}$
C_i	Nœuds constituant l'ensemble des parents de la variable X_i
r_i	Nombre d'états de la variable X_i
q_i	Nombre d'états des parents de la variable X_i.
$\theta_{i,j,k}$	La probabilité que le nœud X_i soit dans l'état k sachant que ses parents sont dans l'état j. $\theta_{i,j,k} = P(X_i = x_i^k \mid C(X_i) = C_i^j)$
D	Base de données
$N_{i,j,k}$	Nombre d'exemples dans la base de données pour lesquels la variableX_i est dans l'état k et ses parents sont dans l'état j.
$N_{i,j}$	Nombre d'exemples dans la base d'exemples pour lesquels les parents de la variable X_i sont dans l'état j.
$f_{i,j,k}$	le rapport $\frac{N_{i,j,k}}{N_{i,j}}$

Définition 33 *:*

Soit $\mathbf{X}$ un vecteur de variables aléatoires discrètes suivant une distribution décrite par la fonction de densité P. On dit alors que $\underline{X}$ suit une distribution multinomiale d'ordre 1.

Propriété 6 *:*

Si X suit une distribution multinomiale d'ordre 1, alors la probabilité d'obtenir la

séquence $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ *est*

$$P_0(D) = \prod_{\mathbf{x} \in X} P(\mathbf{x})^{N(\mathbf{x})} \tag{3.20}$$

avec $\sum_{\mathbf{x} \in \mathbf{X}} N(\mathbf{x}) = N$

en effet comme les tirages sont indépendants, cette probabilité est égale à

$$P_0(D) = \prod_{u=1}^{N} P(\mathbf{x_u}) \tag{3.21}$$

en réordonnant ce produit par rapport aux différents états possibles $\mathbf{x}$ du vecteur $\mathbf{X}$ on aura l'équation 3.20

Définition 34 *:*
Soit $\mathbf{X}$ *un vecteur de variables aléatoires suivant une distribution multinomiale d'ordre 1 de fonction de densité* $\mathbf{P}$*.*
La fonction de vraisemblance de la table de probabilités $\mathbf{P}$ *fondée sur N observations de X est égale à la probabilité d'obtenir cet ensemble d'observations dans l'ordre.*

$$L(\mathbf{P}, N) = \prod_{\mathbf{x} \in \mathbf{X}} \mathbf{P}(\mathbf{x})^{N(\mathbf{x})} \tag{3.22}$$

La log-vraisemblance de la table de probabilité $\mathbf{P}$ *est égale au du logarithme de cette fonction de vraisemblance.*

$$l(P, N) = \sum_{\mathbf{x} \in \mathbf{X}} N(\mathbf{x}) \log(\mathbf{P}(\mathbf{x})) \tag{3.23}$$

Remarque 17 *:*
La vraisemblance mesure en quelque sorte le fait que la séquence d'observations effectuées est vraisemblable dans le cadre de la distribution $\mathbf{P}$*.*

Définition 35 *:*
Soient f et g deux fonctions de densité pour une variable aléatoire discrète ou continue X.
La divergence de Kullback-Leibler entre ces deux fonctions est ;

$$K(f, g) = E_f(\log \frac{f(x)}{g(x)}) \tag{3.24}$$

Dans le cas d'une variable aléatoire discrète, cette fonction se définit par

$$K(f, g) = \sum_{x \in X} f(x) \log \frac{f(x)}{g(x)} \tag{3.25}$$

Dans le cas d'un vecteur de variables aléatoires discrètes, la mesure de la divergence se généralise immédiatement à :

$$K(F,G) = \sum_{\mathbf{x}\in\mathbf{X}} F(\mathbf{x}) \log \frac{F(\mathbf{x})}{G(\mathbf{x})} \tag{3.26}$$

Remarque 18 *:*
La mesure de Kullback-Leibler est dissymétrique en f et g.
elle est définie positive c-a-d. strictement positive pour $f \neq g$ et nulle pour $f = g$.

Propriété 7 *:*
En notant $\mathbf{F} = \frac{\mathbf{N}}{N}$ la distribution de probabilité sur le vecteur $\mathbf{X}$ obtenue comme la mesure des fréquences de chaque réalisation x dans la base d'exemples, la log-vraisemblance peut être définie par

$$l(\mathbf{P}, N) = l(\mathbf{F}, \mathbf{N}) - N.K(\mathbf{F}, \mathbf{P}) \tag{3.27}$$

Remarque 19 *:*
Comme la KL-divergence est positive, l'équation 3.27 montre que la vraisemblance croît quand la KL-divergence décroît.
Le maximum de la vraisemblance est atteint quand $\mathbf{P} = \mathbf{F}$; autrement dit quand la distribution de probabilité est égale à la distribution empirique (fréquences observées).

Définition 36 *:*
Un vecteur de variables aléatoires $(X_1, X_2, ..X_m)$ suit une distribution de Dirichlet de paramètre $\alpha = (\alpha_k)_{1\leq k\leq m}$ si sa densité de probabilité est telle que

$$f_\alpha(u) = \frac{\Gamma(\sum_{j=1}^{m} \alpha_j)}{\prod_{j=1}^{m} \Gamma(\alpha_j)} \prod_{j=1}^{m} u_j^{\alpha_j - 1} \quad 0 < u_j < 1, \sum_{j=1}^{m} u_j = 1. \tag{3.28}$$

on peut dire aussi que

$$f_\alpha(u) \propto \prod_{j=1}^{m} u_j^{\alpha_j - 1}$$

3.6.3 Approche statistique dans le cas d'une structure connue et des données complètes

L'outil de base de l'approche statistique c-a-d la recherche du maximum de vraisemblance [Bickel et Doksum(2001)] s'applique donc directement aux réseaux bayésiens.

3.6.3.1 Apprentissage statistique des paramètres

On sait déjà que le maximum de la vraisemblance de la distribution P est atteint quand $P = \mathbf{F}$. (remarque 19)
Cependant l'utilisation d'un réseau Bayésien pour présenter P fait une hypothèse supplémentaire, à savoir que P se factorise suivant une structure particulière S du réseau. Il faut donc rechercher l'ensemble des paramètres Θ qui maximise la log-vraisemblance dans le cadre de la structure S.
Soit $E^u = (x_1^{k_1(u)}, x_2^{k_2(u)}, ...x_n^{k_n(u)})$ une observation de l'ensemble des variables du réseau.
En appliquant la définition du réseau Bayésien, la probabilité d'observer cette observation dans la distribution représentée par P est :

$$P(\{x_1^{k_1(u)}, x_2^{k_2(u)}, ..., x_n^{k_n(u)}\}) = \prod_{i=1}^{n} P(X_i = x_i^{k_i(u)} \mid C(X_i) = C_i^{j(u)}) = \prod_{i=1}^{n} \theta_{i,j_i(u),k_i(u)} \tag{3.29}$$

La vraisemblance P par rapport à la base de données D est égale au produit des $\theta_{i,j_i(u),k_i(u)}$ pour toutes les observations E^u.
Un terme $\theta_{i,j,k}$ apparaîtra dans ce produit chaque fois qu'une observation E^u présentera la configuration $j_i(u) = j$ et $k_i(u) = k$. On a alors :

$$L(P, N) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} (\theta_{i,j,k})^{N_{i,j,k}} \tag{3.30}$$

La log-vraisemblance de L par rapport à la base de données est donc une fonction de Θ qui peut s'écrire sous la forme

$$\Phi_D(\Theta) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{i,j,k} \log(\theta_{i,j,k}) \tag{3.31}$$

En remarquant que tous les $\theta_{i,j,k}$ sont indépendants, hormis les relations :

$$\sum_{k=1}^{r_i} \theta_{i,j,k} = 1$$

Ainsi on peut réduire le problème de recherche du vecteur Θ qui maximise Φ_D à un ensemble de sous-problèmes indépendants de maximisation sous contraintes.
On note

$$\begin{aligned} \Theta_{i,j} &= (\theta_{i,j,k})_{k=1,r_i} \\ \Phi_D^{i,j}(\Theta) &= \sum_{k=1}^{r_i} N_{i,j,k} \log(\theta_{i,j,k}) \end{aligned} \tag{3.32}$$

Le problème se ramène alors à rechercher les vecteurs $\Theta_{i,j}^{MV}$ tels que :

$$\Theta_{i,j}^{MV} = \arg\max_{\Theta_{i,j}}(\Phi_D^{i,j}) \tag{3.33}$$

de plus on peut écrire

$$\Phi_D^{i,j}(\Theta) = \sum_{k=1}^{r_i-1} N_{i,j,k}\log(\theta_{i,j,k}) + N_{i,j,r_i}\log\left(1-\sum_{k=1}^{r_i-1}\theta_{i,j,k}\right) \tag{3.34}$$

la nullité de $\dfrac{\partial\Phi_D^{i,j}}{\partial\theta_{i,j,k}}$ pour $k = 1, r_i$ entraîne

$$\theta_{i,j,k}^{MV} = \frac{N_{i,j,k}}{N_{i,j}} \tag{3.35}$$

Ainsi on a montré que si la base de données est complète alors le vecteur de paramètres Θ^{MV} qui rend l'observation de cette base la plus probable sous l'hypothèse de la structure S, est le vecteur obtenu à partir des fréquences observées $f_{i,j,k}$.

3.6.4 Approche bayésienne dans le cas d'une structure fixée et des données complètes

L'utilisation du théorème de Bayes permet de proposer une formulation bayésienne de l'apprentissage (plus générale). Cette méthode s'applique surtout si on a le nombre de paramètres très élevé par rapport au nombre de données d'entraînement.

3.6.4.1 Probabilités des modèles

Soient X un ensemble de variables et D une base de données regroupant les observations de ces variables. Soit $M \in \mathcal{M}$ où $\mathcal{M}$ est une famille de modèles.
Supposons que $P(M, D)$ ait un sens c-a-d. que le modèle M puisse être représenté par un ensemble de paramètres θ, et qu'on puisse définir une distribution de probabilité sur ces paramètres. On applique la règle de Bayes [Lauritzen *et al.*(1999)] on a alors :

$$P(M \mid D) = \frac{P(D \mid M).P(M)}{P(D)} \tag{3.36}$$

et aussi

$$P(D) = \int_M P(D, M).dP(M) \tag{3.37}$$

l'expression $P(D \mid M)$ est une fonction de M , elle est appelée la vraisemblance et $P(M \mid D)$ est nommée la distribution a posteriori pour une base de données et une distribution a priori $P(M)$, $P(D)$ est une constante indépendante de M. donc

$$P(M \mid D) \propto P(D \mid M).P(M) \tag{3.38}$$

qui est la formule fondamentale de l'apprentissage Bayésien, et que l'on peut lire

A Posteriori = Vraisemblance . A Priori

3.6.4.2 Apprentissage de modèle par maximum a posteriori

L'apprentissage par *maximum a posteriori* (MAP) [Jordan et Weiss(2001)] revient à chercher le modèle dont la probabilité a posteriori, c-a-d ayant observé la base d'exemples D, est maximale :

$$M^{MAP} = \arg\max_{M} P(M \mid D) \tag{3.39}$$

Dans le cas où l'hypothèse effectuée sur la distribution a priori sur M est celle de la distribution uniforme, l'apprentissage MAP est équivalent à l'apprentissage par maximum de vraisemblance [Becker et Naim(1999)].

3.6.4.3 Apprentissage Bayésien de paramètres

Étant donné une structure fixe, le principe de l'apprentissage Bayésien peut être appliqué directement à l'apprentissage de paramètres. Soit B un BN de structure fixée et de paramètres Θ. Soit $P(\Theta, X_1, X_2, ..., X_n)$ la loi jointe des paramètres Θ et des variables $(X_i)_{i=1,n}$. Soit D une base de données associée à B. La formule de l'apprentissage Bayésien s'écrit :

$$P(\Theta \mid D) \propto P(D \mid \Theta).P(\Theta) \tag{3.40}$$

On sait que

$$P(D \mid \Theta) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} (\theta_{i,j,k})^{N_{i,j,k}} \tag{3.41}$$

d'où

$$P(\Theta \mid D) \propto \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} (\theta_{i,j,k})^{N_{i,j,k}}.P(\Theta) \tag{3.42}$$

L'apprentissage MAP consiste à rechercher l'ensemble de paramètres Θ dont la probabilité a posteriori c-a-d. sachant que D a été observée, est maximale :

$$\Theta^{MAP} = \arg\max_{\Theta}(\prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} (\theta_{i,j,k})^{N_{i,j,k}}.P(\Theta)) \tag{3.43}$$

Dans le cas où l'on ne fait aucune hypothèse a priori sur Θ, autrement dit dans le cas où $P(\Theta)$ est uniforme, la recherche de Θ est équivalent à la recherche du MV ce qui est conforme à la formule générale de l'apprentissage Bayésien.
Ce résultat en particulier justifie que certains chercheurs [Heckerman *et al.*(1994)]

aient proposé d'utiliser une distribution de Dirichlet comme distribution a priori des paramètres Θ. donc

$$P(\Theta) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} (\theta_{i,j,k})^{\alpha_{i,j,k}} \tag{3.44}$$

Ainsi la distribution a posteriori de Θ s'écrit comme une distribution de Dirichlet :

$$P(\Theta \mid D) \propto \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} (\theta_{i,j,k})^{N_{i,j,k}+\alpha_{i,j,k}} \tag{3.45}$$

Ainsi on peut dire que la famille des distributions de Dirichlet est une famille conjuguée pour le tirage par une distribution multinomiale. Cela est avantageux car l'observation de nouveaux exemples ne change pas la forme de la distribution des paramètres Θ, et de plus cela permet de considérer l'apprentissage de façon totalement incrémentale. On peut remarquer que l'équation 3.45 a la même forme que l'équation 3.42 . La recherche des paramètres Θ réalisant le MAP peut donc s'interpréter comme la recherche des paramètres Θ donnant le maximum de vraisemblance pour une base d'exemples où les comptages seraient :

$$N'_{i,j,k} = N_{i,j,k} + \alpha_{i,j,k} \tag{3.46}$$

La solution du problème dans le cadre de l'apprentissage par MAP est donc :

$$\theta^{MAP}_{i,j,k} = \frac{N_{i,j,k} + \alpha_{i,j,k}}{N_{i,j} + \alpha_{i,j}} \tag{3.47}$$

ou

$$\alpha_{i,j} = \sum_{k} \alpha_{i,j,k}$$

Autrement dit les paramètres $\alpha_{i,j,k}$ peuvent être considérés comme des hypothèses a priori sur les observations.

3.6.5 Apprentissage avec des données incomplètes

Dans le cas de données manquantes c-a-d. où certaines variables ne sont pas observées pour certains exemples, les calculs développés précédemment ne fonctionnent plus car la log-vraisemblance de la base d'exemples observée [équation 3.41] ne se factorise plus puisqu'on ne peut pas exprimer la vraisemblance d'un exemple partiellement observé, comme un terme $\theta_{i,j,k}$. Et donc la recherche de paramètres ne peut pas s'effectuer de façon directe. La seule solution est de compléter les cas manquants avec les valeurs les plus probables. Nous allons montrer comment on peut maximiser la log-vraisemblance en utilisant deux méthodes différentes : *le gradient croissant* et

la méthode *EM* Soit $D = (D_m)_m$ une base d'exemples.
La log-vraisemblance est donnée par

$$\begin{aligned} L &= \log(P(D \mid M) \\ &= \sum_m \log(P(D_m \mid M) \\ &= \sum_m \log(P_\theta(D_m)) \end{aligned} \tag{3.48}$$

3.6.5.1 Méthode de descente du gradient

Une méthode pour maximiser la log-vraisemblance est d'utiliser le gradient croissant [Heckerman *et al.*(1994), Binder *et al.*(1997)]. Nous allons montrer que le gradient croissant est la somme des marginales qui peuvent être calculées en utilisant un moteur d'inférence.
On rappelle

$$\theta_{i,j,k} = P(X_i = k \mid C(X_i) = j) \quad \text{avec} \sum_k \theta_{i,j,k} = 1 \quad \forall i, j$$

On a

$$\begin{aligned} \frac{\partial L}{\partial \theta_{i,j,k}} &= \sum_m \frac{\partial \log(P_\theta(D_m))}{\partial \theta_{i,j,k}} \\ &= \sum_m \frac{\partial P_\theta(D_m)}{\partial \theta_{i,j,k}} \frac{1}{P_\theta(D_m)} \end{aligned} \tag{3.49}$$

et pour

$$\begin{aligned} \frac{\partial P_\theta(D_m)}{\partial \theta_{i,j,k}} &= \frac{\partial}{\partial \theta_{i,j,k}} \sum_{j',k'} P_\theta(D_m \mid X_i = k', C(X_i) = j') P_\theta(X_i = k' \mid C(X_i) = j') P_\theta(C(X_i) = j') \\ &= P_\theta(D_m \mid X_i = k, C(X_i) = j).P_\theta(C(X_i) = j) \end{aligned} \tag{3.50}$$

Ainsi

$$\begin{aligned} \frac{\partial P_\theta(D_m)}{\partial w_{i,j,k}} \frac{1}{P_\theta(D_m)} &= \frac{P_\theta(D_m \mid X_i = k, C(X_i) = j).P_\theta(C(X_i) = j)}{P_\theta(D_m)} \\ &= \frac{P_\theta(X_i = k, C(X_i) = j \mid D_m).P_\theta(D_m) P_\theta(C(X_i) = j)}{P_\theta(X_i = k, C(X_i) = j).P_\theta(D_m)} \\ &= \frac{P_\theta(X_i = k, C(X_i) = j \mid D_m)}{\theta_{i,j,k}} \end{aligned} \tag{3.51}$$

Ainsi à partir de l'équation 3.51, le calcul du gradient croissant devient un simple calcul en utilisant un moteur d'inférence.

3.6.5.2 EM

La méthode la plus générale de prise en compte des données incomplètes est fondée sur un algorithme de recherche itérative du modèle, inspiré de l'algorithme EM. Cet algorithme est détaillé dans l'annexe C.

Chapitre 4

Réseaux Bayésiens dynamiques et applications

4.1 Introduction

Ces dernières années, dans plusieurs domaines, l'observation d'un environnement se fait à travers un flux d'information appelé *données séquentielles*. Ces données peuvent être des séries chronologiques générées par un système dynamique, ou une séquence générée par un processus spatial à une dimension. Il est possible d'analyser ces données en temps réel : analyse en ligne, ou après leur collecte. Dans le cas d'analyse en ligne, on prédit les observations futures connaissant l'ensemble des observations jusqu'au temps courant. On note $y_{1:t} = (y_1, ..., y_t)$ la séquence d'observation jusqu'au temps t (le temps est supposé discret). Prédire l'état futur (phénomène incertain) revient donc à estimer la distribution de probabilité $P(y_{t+\tau} \mid y_{1:t})$ à l'instant $t + \tau$ où τ est l'horizon i.e la distance à laquelle on veut prédire dans le futur. Parfois on peut avoir un certain contrôle du système qu'on surveille (l'observateur peut agir sur l'environnement), ces actions peuvent être prises en compte sous forme d'une commande u appelée entrée. On note $u_{1:t}$ les entrées depuis l'instant initial jusqu'à l'instant courant t et $u_{t+1:t+\tau}$ les τ prochaines entrées. Ainsi prédire l'état futur revient à calculer la distribution de probabilité $P(y_{t+\tau} \mid u_{1:t+\tau}, y_{1:t})$. Les approches classiques pour résoudre ce problème utilisent les modèles linéaires [Hamilton(1994)], ou les modèles non-linéaires [Meek *et al.*(2002)]. Pour les données discrètes les modèles n-grammes sont couramment utilisés [Jelinek(1997)]. Ces approches font apparaître quelques problèmes :

- la prédiction doit être basée sur une fenêtre de temps finie $y_{t-k:t}$ où $k \geq 0$ est la taille de la fenêtre temporelle. Si le système modélisé est markovien avec un ordre inférieur ou égal à l, alors la prédiction avec une fenêtre $y_{t-l:t}$ plus

petite que l'historique $y_{1:t}$ complet donnera les mêmes résultats. Mais dans de nombreux cas, l'ordre est inconnu et grand.

- il est difficile d'incorporer la connaissance a priori dans une approche classique : beaucoup de connaissances ne peuvent pas s'exprimer en termes de quantités observables.

Pour résoudre ces problèmes, nous allons utiliser les modèles d'espace d'états. On suppose que l'environnement observé possède un certain nombre d'états cachés qui gèrent les observations et que ces états cachés évoluent au cours du temps. Ainsi le but est d'inférer les états cachés sachant les observations jusqu'au temps présent. Si X_t est l'état caché à l'instant t, alors l'inférence sera l'estimation de la distribution de probabilité $P(X_t \mid y_{1:t}, u_{1:t})$. Ce dernier est appelé l'état de croyance. L'état de croyance peut être mis à jour récursivement en utilisant la règle de Bayes. Pour chaque modèle d'espace, il faut définir l'état initial $P(X_1)$, la fonction de transition d'état $P(X_t \mid X_{t-1})$ et la fonction de l'observation $P(y_t \mid X_t)$. Dans la plupart des cas on suppose que le modèle est markovien du premier ordre :

$$P(X_t \mid X_{1:t}) = P(X_t \mid X_{t-1})$$

et que les fonctions de transition et d'observation sont les mêmes au cours du temps (modèle homogène). Parmi les différentes approches des modèles d'espace d'états, on décrit ici les réseaux bayésiens dynamiques.

4.2 Représentation d'un réseau Bayésien dynamique

Les réseaux bayésiens dynamiques (DBN) [Murphy(2002)] sont une extension des réseaux bayésiens qui représente l'évolution temporelle de variables aléatoires. Ainsi un réseau bayésien dynamique est une chaîne d'un même réseau bayésien (sous-réseau) répété autant de fois que nécessaire (suivant la longueur de la séquence d'observations). On note que le terme dynamique signifie qu'on modélise un système dynamique mais pas le changement du réseau en fonction du temps t. Plus formellement :

Définition 37 *:*
On considère un ensemble $X_t = (X_t^1, ..., X_t^N)$ de variables évoluant dans le temps [1, T]. Un DBN est défini par le couple (B_1, B_2), avec

- *B_1 est un BN qui définit la probabilité a priori $P(X_1)$ (état initial).*
- *B_2 est le réseau temporel à deux pas de temps [2TBN : two-slice temporal Bayes net] , qui définit $P(X_t \mid X_{t-1})$ à l'aide d'un graphe acyclique orienté*

(DAG) de la façon suivante :

$$P(X_t \mid X_{t-1}) = \prod_{i=1}^{N} P(X_t^i \mid C(X_t^i)) \tag{4.1}$$

où X_t^i est le i ème nœud au temps t, et $C(X_t^i)$ est le parent de X_t^i dans le graphe.

Il est important de remarquer que les parents d'un nœud X_t^i peuvent être soit dans le même pas de temps soit dans le pas de temps précédent.

Chaque nœud du deuxième pas de temps 2TBN a une distribution de probabilité conditionnelle associée (CPD) qui définit $P(X_t^i \mid C(X_t^i))$ pour $t > 1$. Ces distributions de probabilité ont des formes différentes (table, mélange de gaussiennes, etc $\cdots$). On suppose toujours que le modèle est markovien d'ordre un. Les arcs reliant les nœuds entre les différents pas de temps (de gauche vers la droite) représentent le déroulement continu du temps. S'il existe un tel arc d'un nœud X_{t-1}^i à un autre X_t^i, on dit alors que ce nœud est *persistant*. Ainsi les arcs entre deux pas de temps sont considérés comme la persistance d'un phénomène au cours du temps et les arcs au sein d'un même pas de temps sont considérés comme un effet causal immédiat. La sémantique du DBN peut être définie par le déroulement du 2TBN jusqu'à la longueur T. La distribution jointe résultante est donnée par

$$P(X_{1:T}) = \prod_{t=1}^{T} \prod_{i=1}^{N} P(X_t^i \mid C(X_t^i)) \tag{4.2}$$

Ceci revient à représenter un réseau bayésien dynamique déroulé comme un réseau bayésien sous forme de chaîne dans lequel les variables sont aussi des réseaux bayésiens statiques.

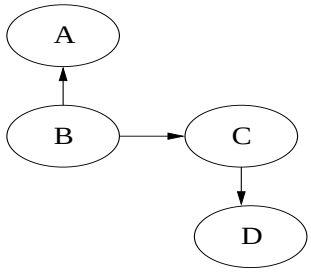


Fig. 4.1 – Un pas de temps (slice) pour un réseau dynamique

Par exemple la figure Fig. 4.1 représente un pas de temps.

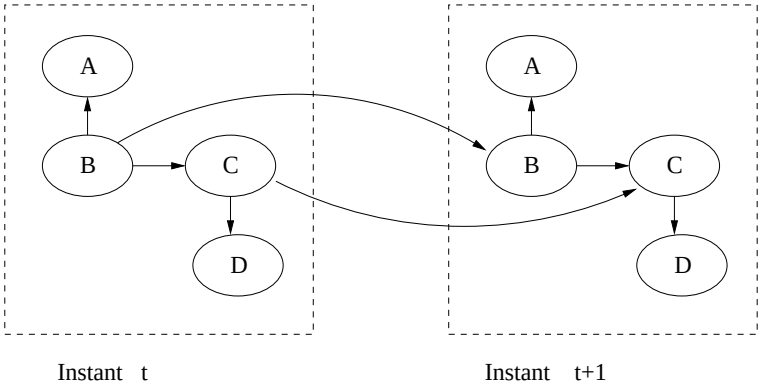


Fig. 4.2 – Un 2TBN avec les liens entre les pas (slice) à l'instant t et t+1

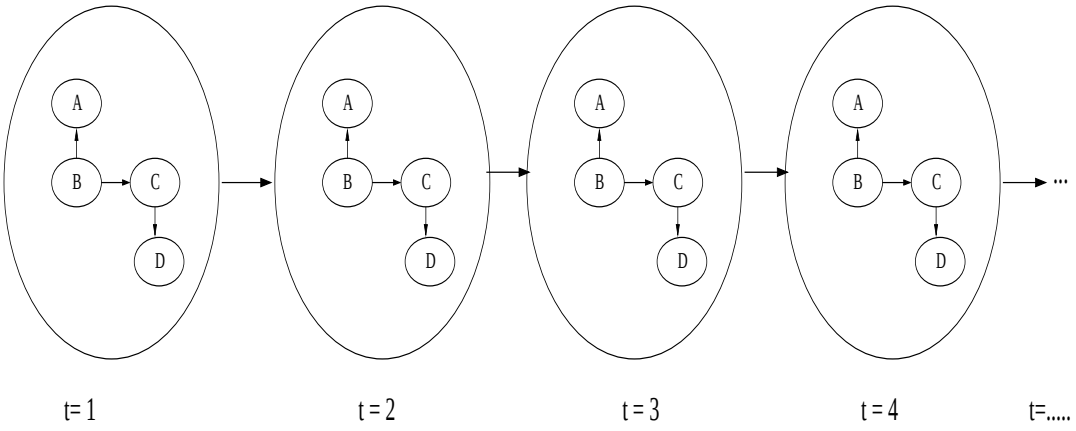


Fig. 4.3 – Un réseau bayésien contenant des réseaux bayésiens identiques locaux dans chaque nœud la flèche représente les liens entre les différents pas de temps

La figure Fig.4.2 représente le 2TBN correspondant au modèle de base. Le réseau bayésien dynamique final est dans la figure.4.3.

La distribution de probabilité associée à chaque nœud du réseau de la figure Fig.4.3 est factorisée au sein d'un réseau bayésien statique (celui de la Fig.4.1). Ainsi au lieu d'utiliser deux réseaux, on peut en créer un seul dans lequel les arcs entre deux nœuds de pas différents représentent les dépendances temporelles entre chaque pas de temps. L'objectif est d'avoir un seul formalisme pour représenter à la fois un réseau statique et un réseau dynamique comme le montre la figure Fig.4.4.

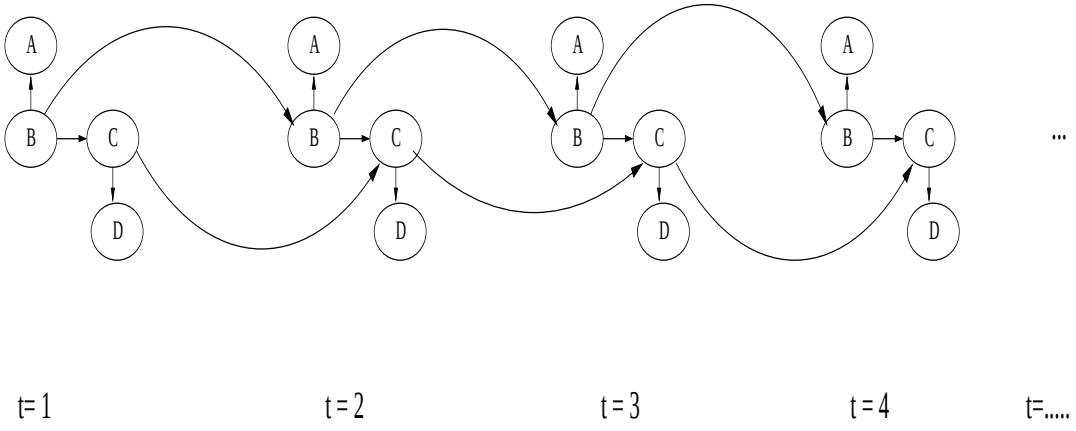


Fig. 4.4 – Un réseau bayésien dynamique déroulé sur plusieurs pas

En ce qui concerne les CPDs on les trouve dans les tableaux Tab.4.1 et Tab.4.2.

Dpa	Cpa	CPD
-	-	$P(Y = j) = \pi(j)$
i	-	$P(Y = j \mid X = i) = A(i, j)$
-	u	$P(Y = j \mid U = u) = \sigma(u, W, j)$
i	u	$P(Y = j \mid U = u, X = i) = \sigma(u, W_i, j)$

Tab. 4.1 – Exemples de CPDs dans le cas d'un fils discret avec : 0 ou 1 parent discret (Dpa), 0 ou 1 parent continu (Cpa)

Dpa	Cpa	CPD
-	-	$P(Y = y) = \mathcal{N}(y, \mu, \Sigma)$
i	-	$P(Y = y \mid X = i) = \mathcal{N}(y, \mu_i, \Sigma_i)$
-	u	$P(Y = y \mid U = u) = \mathcal{N}(y, Wu + \mu, \Sigma)$
i	u	$P(Y = y \mid U = u, X = i) = \mathcal{N}(y, W_i u + \mu_i, \Sigma_i)$

Tab. 4.2 – Exemples de CPDs dans le cas d'un fils continu avec : 0 ou 1 parent discret (Dpa), 0 ou 1 parent continu (Cpa)

où
- W est la matrice de régression.
- σ est une généralisation de la fonction sigmoïde (logistique). Elle est donnée par la formule suivante :

$$\sigma(u, W, j) = P(Y = j \mid U = u) = \frac{\exp((u^t.W)_j)}{\sum_k \exp((u^t.W)_k)}$$

On considère généralement que le processus est stationnaire c-a-d., que les hypothèses d'indépendance et les probabilités conditionnelles associées sont identiques pour tous les temps t. Dans ce cas un DBN peut être représenté par un BN dont la structure est dupliquée pour chaque pas de temps t.

Remarque 20 *La différence entre un DBN et un HMM est que le DBN représente l'état caché en terme d'ensembles de variables aléatoires cachées $X_t^1, ..., X_t^N$, en revanche dans un HMM, l'espace d'état se compose d'une variable aléatoire simple X_t.*

4.3 Représentation des HMMs et de leurs variantes comme DBNs

On peut représenter un HMM comme un DBN, un nœud X_t (resp.Y_t) est une variable aléatoire dont la valeur indique l'état occupé (resp. l'observation) au temps t (Fig.4.3). Le temps apparaît de façon explicite et les flèches qui lient les X_t indiquent les dépendances et non pas les transitions entre états. Un HMM peut être défini par le premier et le deuxième pas de temps (slice), puisqu'on a une répétition de la structure, ainsi que le critère de l'indépendance conditionnelle (ou d-séparation) :

$$X_{t+1} \perp X_{t-1} \mid X_t \quad et \quad Y_t \perp Y_{t'} \mid X_t \quad pour \quad t' \neq t. \tag{4.3}$$

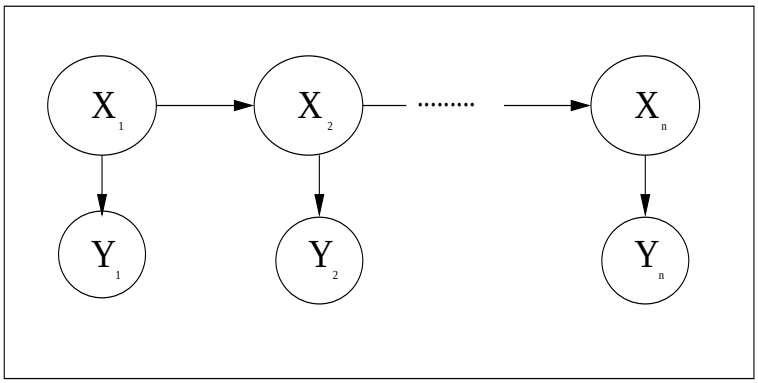


FIG. 4.5 – HMM représenté comme un exemple de DBN, ce modèle peut être défini par la seule connaissance des deux premiers pas de temps

Pour les BNs on doit définir la distribution de la probabilité conditionnelle (CPD) pour chaque nœud sachant ses parents. Donc pour un HMM, modélisé par un DBN, celui ci est entièrement défini par la donnée des paramètres suivants :

$$\begin{cases} P(X_1) \\ P(X_t \mid X_{t-1}) \\ P(Y_t \mid X_t) \end{cases} \tag{4.4}$$

La CPD pour $P(X_1)$ est en général représentée par un vecteur :

$$P(X_1 = i) = \pi(i) \quad avec \quad 0 \leq \pi(i) \leq 1 \quad et \quad \sum_i \pi(i) = 1.$$

La probabilité $P(X_t \mid X_{t-1})$ est souvent représentée par une matrice stochastique A (matrice de transition) ie.

$$A(i,j) = P(X_t = j \mid X_{t-1} = i) \quad \text{telle que} \quad \sum_j A(i,j) = 1 \qquad \forall i.$$

Pour $P(Y_t \mid X_t)$, il y a deux cas

– soit les variables Y_t sont discrètes, et dans ce cas

$$B(i,j) = P(Y_t = j \mid X_t = i)$$

– soit les Y_t sont continues , et dans ce cas on modélise classiquement la distribution par un mélange de gaussiennes

$$P(Y_t = j \mid X_t = i) = \sum_{m=1}^{M_i} C_{i,m} \eta(y_t, \mu_{i,m}, \Sigma_{i,m})$$

où

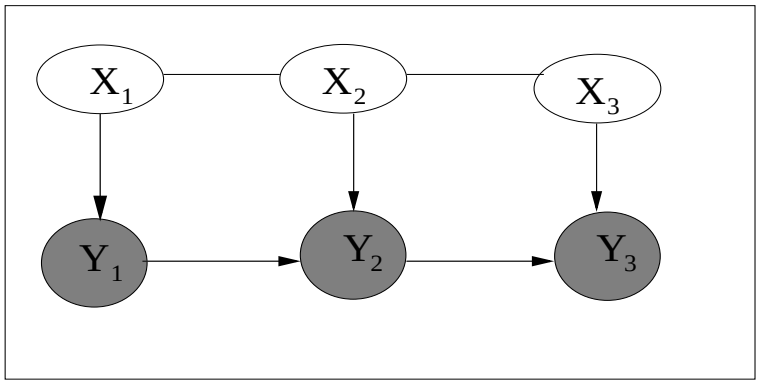


Fig. 4.6 – Un HMM auto-régressif (modèle de trajectoire)

- M_i est le nombre de mélange de gaussiennes dans l'état i.
- $C_{i,m}$ est le poids de la m^{eme} composante.
- $\mathcal{N}(., \mu, \Sigma)$ est une gaussienne de moyenne μ et de matrice de variance covariance Σ telle que

$$\mathcal{N}(y, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n \mid \Sigma \mid}} e^{\frac{-1}{2}(y-\mu)'\Sigma^{-1}(y-\mu)}$$

Si on suppose que tous les paramètres sont invariants par rapport au temps, on aura besoin seulement de $P(X_1)$, $P(X_2 \mid X_1)$ et $P(Y_1 \mid X_1)$. Les CPDs des futurs pas de temps (slices) sont les mêmes que celles du 2^{eme} pas de temps, ce qui réduit beaucoup la quantité des données demandées pour l'apprentissage.

4.3.1 HMMs auto-régressifs (modèles de trajectoire)

L'hypothèse standard des HMMs :

$$Y_t \perp Y_{t'} \mid X_t \qquad \forall t \in [0 \quad T] \tag{4.5}$$

est une hypothèse forte, on peut l'affaiblir en liant les observations Y_t, Y_{t+1} entre elles (voir Fig.4.6). Ce qui nous donne un modèle appelé *HMM auto-régressif* (AR-HMM). Il permet de réduire l'effet des états X_t en permettant aussi bien à Y_{t-1} d'aider à prévoir Y_t. Ceci conduit souvent à des modèles avec une vraisemblance plus élevée. Si Y_t est discrète, la CPD de Y_t peut être représentée comme une table. Si Y_t est continue, la CPD de Y_t peut être représentée par

$$P(Y_t = y_t \mid X_t = i, Y_{t-1} = y_{t-1}) = \mathcal{N}(y_t, W_i y_{t-1} + \mu_i, \Sigma_i) \tag{4.6}$$

où W_i est la matrice de régression associée à l'état $X_t = i$. De plus ce modèle est connu sous le nom de HMM de corrélation ([Hamilton(1990)]).

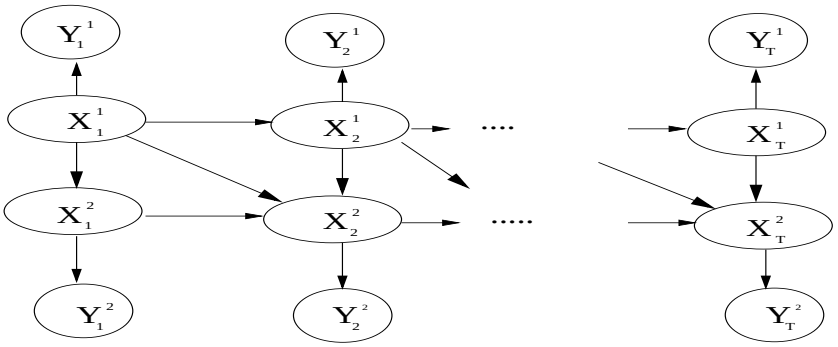


FIG. 4.7 – Couplage de deux HMMs

4.3.2 HMMs couplés : CHMMs

Dans un HMM couplé [Saul et Jordan(1995), Brand(1997)]), on suppose que chaque variable (nœud) cachée a sa propre observation (variable observable), et qu'elle est aussi liée à d'autres variables nœuds observables ou cachés (voir la figure Fig.4.7). Nous allons utiliser les CHMMs (HMMs couplés) pour coupler deux HMMs l'un appelé HMM-vertical et l'autre HMM-horizontal. Notre modèle est entièrement décrit par la donnée des paramètres suivants :

$$\begin{cases} A^1_{i,j} = P(X^1_t \mid X^1_{t-1}) \qquad t \geq 2 \\ U_{i,j,k,l} = P(X^2_t = k \mid X^2_{t-1} = i, X^1_t = j, X^1_{t-1} = l) \qquad t \geq 2 \\ b^l_i(.) = P(Y^l_t = . \mid X^l_t = i) \quad \text{pour} \quad l = 1, 2 \end{cases} \tag{4.7}$$

Dans ce cas on remarque que nous utilisons plus de paramètres et en outre nous pouvons avoir d'autres types de couplage [Hallouli *et al.*(2003)] .

4.4 Inférence dans les DBNs

Le but de l'inférence dans un DBN est de calculer $P(X^i_t \mid Y_{1:\tau})$ où τ est la longueur de la séquence d'observation. Ceci revient à estimer les états cachés en utilisant uniquement les observations. Il existe plusieurs sortes d'inférence possibles :

- **Filtrage** : elle consiste à estimer l'état de croyance en utilisant la règle de Bayes à partir des observations jusqu'à l'instant t, ie :

$$P(X_t \mid Y_{1:t}) \propto P(Y_t \mid X_t, Y_{1:t-1}) P(X_t \mid Y_{1:t-1})$$

On remarque qu'en utilisant la propriété de Markov on aura

$$P(Y_t \mid X_t, Y_{1:t-1}) = P(Y_t \mid X_t)$$

- **Prédiction** : dans ce cas il faut estimer un état futur sachant les observations jusqu'à l'instant courant. Ceci revient à calculer $P(X_{t+h} \mid Y_{1:t})$ où $h > 0$.
- **Lissage à intervalle régulier** : il consiste à estimer un état passé sachant les observations jusqu'à l'instant courant. Ceci revient à calculer $P(X_{t-l} \mid Y_{1:t})$ où $l > 0$.
- **Analyse a posteriori** : il s'agit d'estimer $P(X_t \mid Y_{1:T})$ pour $1 \leq t \leq T$. Il s'agit donc d'un lissage mais que l'on effectue après avoir obtenu l'ensemble des observations.

Remarque 21 *:*
Les techniques de l'apprentissage dans un réseau bayésien dynamique sont une simple extension des techniques d'apprentissage traitées dans le chapitre précédent.

4.5 Applications

Dans cette partie nous allons tester nos algorithmes comme précédemment en utilisant la base de caractères : **UW English Document Image Database** et aussi la base de chiffres **MNIST**.
A partir de la base **UW English**, nous avons construit trois parties :

- **base d'apprentissage** : est constituée de 6240 exemples de caractères.
- **base de validation** : est constituée de 4160 exemples de caractères.
- **base de test** : est constituée de 4160 exemples de caractères.

En ce qui concerne la base **MNIST**, nous avons construit aussi trois parties :

- **base d'apprentissage** : est constituée de 40000 chiffres.
- **base de validation** : est constituée de 20000 chiffres.
- **base de test** : est constituée de 10000 chiffres.

4.5.1 Outil Bayesnet

Bayesnet est une boîte à outil (Toolbox) écrite par Kevin Murphy en Matlab [Murphy(2003)]. Cette Toolbox supporte plusieurs points :

- l'inférence exacte et approchée.
- l'apprentissage des paramètres et des structures dans le cas où toutes les variables sont observables.
- la création des modèles dynamiques et statiques.

Cependant cette Toolbox a des limitations concernant les nombres des nœuds (variables aléatoires), l'impossibilité pour une variable discrète d'avoir des parents

continus, et le temps de calcul est en général important. Nous utilisons pour nos applications cette boîte à outils (version 2002) qui est toujours en cours de développement et en amélioration constante.

4.5.2 Applications aux Modèles HMM sous forme DBN en utilisant la base UW ENGLISH

De même qu'au chapitre 2 nous allons construire deux HMMs nommés *HMM-vertical* et *HMM-horizontal* à l'aide des réseaux bayésiens dynamiques. Le *HMM-vertical* est le modèle qui prend pour séquence d'entrée les colonnes de pixels du caractère. Le *HMM-horizontal* est le modèle qui prend pour séquence d'entrée les lignes de pixels du caractère.
Nous signalons que toutes les transitions entre états sont ici possibles, ie. la matrice des transitions est pleine (type ergodique), contrairement à l'hypothèse gauche-droite faite avec HTK pour les HMMs (Chap.2).
D'autre part la base de validation sert à ajuster tous les paramètres des modèles. Sur cette base de validation nous avons effectué plusieurs expériences pour trouver le nombre d'états optimal pour chaque modèle. Nous procédons de la façon suivante : nous cherchons le nombre optimal d'états en supposant au départ que tous les modèles de caractère ont le même nombre d'états (de la même façon que dans le cas des HMMs en utilisant HTK). Puis nous affinons ce nombre d'états pour chaque modèle : nous recherchons le nombre d'états optimal pour le premier modèle, celui des autres restant fixé. Après avoir trouvé ce nombre optimal, nous passons au deuxième modèle et nous cherchons son nombre d'états optimal, puis nous passons au suivant jusqu'au dernier modèle. Cette procédure itérative a été appliquée pour les majuscules, les minuscules et l'ensemble majuscules minuscules pour chacun des deux HMMs vertical et horizontal. De plus ces nombres optimaux d'états obtenus ici sur des modèles plus simples, seront utilisés par la suite pour les modèles couplés.

Majuscules :

Nous n'utilisons que la partie de la base concernant les majuscules. Dans ce cas le nombre d'états optimal par classe pour les deux modèles : HMM-vertical et horizontal, est dans les tableaux suivants Tab.4.3 et Tab.4.4

modèle	A	B	C	D	E	F	G	H	I	J	K	L	M
nbre d'états	22	14	10	10	22	20	10	12	10	10	12	10	16
modèle	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
nbre d'états	22	10	14	12	12	12	10	14	16	10	10	10	14

Tab. 4.3 – Nombre optimal d'états par classe pour les majuscules : pour le HMM-vertical

modèle	A	B	C	D	E	F	G	H	I	J	K	L	M
nbre d'états	22	16	12	10	20	20	10	12	12	12	12	10	16
modèle	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
nbre d'états	20	10	14	14	12	14	10	16	16	18	16	12	16

Tab. 4.4 – Nombre optimal d'états par classe pour les majuscules : pour le HMM-horizontal

Minuscules :

Ici nous n'utilisons que la partie de la base concernant les minuscules.

Dans ce cas les nombres optimaux d'états sont dans les tableaux suivants Tab.4.5 et Tab.4.6

modèle	a	b	c	d	e	f	g	h	i	j	k	l	m
nbre d'états	16	12	10	10	12	10	12	10	10	10	12	10	14
modèle	n	o	p	q	r	s	t	u	v	w	x	y	z
nbre d'états	12	10	10	12	10	12	10	14	14	18	12	12	12

Tab. 4.5 – Nombre optimal d'états par classe pour les minuscules : pour le HMM-vertical

modèle	a	b	c	d	e	f	g	h	i	j	k	l	m
nbre d'états	18	14	10	12	10	10	14	12	10	10	14	10	14
modèle	n	o	p	q	r	s	t	u	v	w	x	y	z
nbre d'états	12	10	10	14	10	16	10	12	12	14	14	12	14

TAB. 4.6 – Nombre optimal d'états par classe pour les minuscules : pour le HMM-horizontal

Majuscules-Minuscules :

En ce qui concerne l'ensemble Majuscules-Minuscules, les nombres optimaux d'états sont dans les tableaux suivants Tab.4.7 et Tab.4.8. Nous signalons que pendant les expériences, nous avons pris le même nombre d'états pour chaque caractère : majuscule et minuscule.

modèle	A-a	B-b	C-c	D-d	E-e	F-f	G-g	H-h	I-i	J-j	K-k	L-l	M-m
nbre d'états	20	14	10	12	20	18	10	12	10	10	14	10	16
modèle	N-n	O-o	P-p	Q-q	R-r	S-s	T-t	U-u	V-v	W-w	X-x	Y-y	Z-z
nbre d'états	14	10	10	16	14	12	12	14	16	18	12	14	12

TAB. 4.7 – Nombre optimal d'états par classe pour l'ensemble Majuscules-Minuscules : pour le HMM-vertical

modèle	A-a	B-b	C-c	D-d	E-e	F-f	G-g	H-h	I-i	J-j	K-k	L-l	M-m
nbre d'états	18	14	10	10	18	16	12	10	10	10	14	10	16
modèle	N-n	O-o	P-p	Q-q	R-r	S-s	T-t	U-u	V-v	W-w	X-x	Y-y	Z-z
nbre d'états	14	10	12	16	14	14	12	14	14	16	10	12	12

TAB. 4.8 – Nombre optimal d'états par classe pour l'ensemble Majuscules-Minuscules : pour le HMM-horizontal

Ces expériences montrent que le nombre d'états optimal varie entre 10 et 22 états. La méthode itérative mise en œuvre ne garantit cependant pas, pour chaque modèle (classe), d'avoir trouvé le nombre d'états globalement optimum car celui ci peut dépendre de l'ordre dans lequel les modèles ont été optimisés.
Après avoir fixé tous les paramètres des modèles, nous passons à l'étape de test. Les résultats sont dans le tableau Tab.4.9.

Ainsi à partir du Tab.4.9, on déduit que le HMM-vertical donne un taux de reconnaissance meilleur que le HMM-horizontal du fait que les colonnes des images

	Modèle	Taux de reconnaissance
Majuscules	HMM-vertical	91.35 %
	HMM-horizontal	89.67 %
Minuscules	HMM-vertical	89.81 %
	HMM-horizontal	86.72 %
Majuscules-Minuscules	HMM-vertical	88.21 %
	HMM-horizontal	86.13 %

Tab. 4.9 – Résultats de reconnaissance pour les HMMs sous forme de DBN

de caractères sont plus discriminantes pour la classification de caractères imprimés ou manuscrits. D'autre part les résultats associés aux minuscules sont moins bons que ceux associés aux majuscules car les minuscules ont des formes complexes. De plus dans le cas de l'ensemble Majuscules-Minuscules les scores diminuent. Cela du fait que nous obtenons plus de confusion pour les caractères de même forme (nous considérons par exemple la reconnaissance du caractère O comme un o est une erreur). Les taux de reconnaissance en utilisant la boîte à outils HTK sont meilleurs que ceux associés à Bayesnet. Or dans le cas mono-flux un DBN est en théorie équivalent à un HMM. Cela peut se justifier par :

- le type de transition choisi : nous avons utilisé le modèle gauche-droite dans le cas des HMM avec HTK et le modèle ergodique dans le cas des DBNs.
- l'initialisation n'est pas la même pour les deux cas : en utilisant HTK ou en utilisant Bayesnet. HTK utilise notamment un algorithme d'initialisation basé sur les k-moyennes (HInit) pour les modèles gauche-droite. Dans le cas des DBNs, nous ne disposons pas d'une telle procédure d'initialisation.
- la boîte à outils Bayesnet est toujours en cours de développement. De plus elle est récente et écrite par un seul chercheur contrairement à la boîte à outils HTK qui a été developpée pendant plusieurs années dans le cadre d'un travail d'équipe.

La différence entre les résultats des deux logiciels s'explique probablement par ces raisons.

4.5.3 Applications aux modèles HMM sous forme DBN en utilisant la base MNIST

Comme précédemment, nous construisons les deux HMMs : HMM-vertical et HMM-horizontal sous forme de réseau bayésien dynamique. A partir de la base

de validation nous avons ajusté le nombre optimal d'états pour chaque modèle en utilisant la même procédure que pour les caractères imprimés.

Ainsi les résultats concernant le nombre optimal d'état, sont dans les tableaux Tab.4.10 et Tab.4.11.

modèle	0	1	2	3	4	5	6	7	8	9
nbre d'états	12	10	13	12	14	14	14	12	13	11

TAB. 4.10 – Nombre optimal d'états par classe pour le HMM-vertical

modèle	0	1	2	3	4	5	6	7	8	9
nbre d'états	12	9	14	15	14	15	13	12	14	12

TAB. 4.11 – Nombre optimal d'états par classe pour le HMM-horizontal

Ainsi les résultats de test sont dans le tableau Tab. 4.12

	Taux de reconnaissance
HMM-vertical	92.85 %
HMM-horizontal	90.17 %

TAB. 4.12 – Résultats de reconnaissance pour les HMMs sous forme DBN (MNIST)

A partir des résultats (4.12), on constate que le HMM-vertical est meilleur que le HMM-horizontal. Nous signalons que le temps de calcul est important dans ce cas. La confusion la plus remarquée est celle du chiffre 4 avec le 9 et 7 et du 5 avec le 8 et 6. Par exemple le chiffre de la figure Fig.4.8 est reconnu comme un 9 pour le premier choix et comme 7 dans le deuxième choix. Nous remarquons aussi que les résultats associés à la boîte à outils HTK sont meilleurs que ceux associés à Bayesnet pour les mêmes raisons citées en 4.5.2 dans le cas de la base UW.

4.5.4 Applications des modèles de trajectoire (auto-régressifs)

Dans ce paragraphe nous utilisons le modèle auto-régressif déjà défini. On définit de la même manière deux types de modèles :

- *Traj-vertical* est un modèle qui prend pour séquence d'entrée (observations) les colonnes de pixels du caractère.
- *Traj-horizontal* est un modèle qui prend pour séquence d'entrée (observations) les lignes de pixels du caractère.

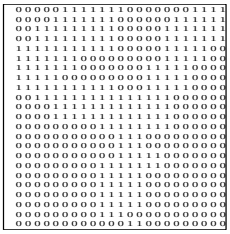


Fig. 4.8 – Chiffre 4

Nous utilisons les deux types de trajectoire pour la reconnaissance en employant les bases : UW et MNIST.

4.5.4.1 Applications des modèles de trajectoire à la base UW

Le nombre optimal d'états pour chaque classe est celui trouvé dans le cas des HMMs simples (cependant on pourrait refaire le même protocole pour la recherche du nombre optimal d'états cachés). Ces modèles de trajectoire prennent en considération les dépendances entre les observations lignes ou colonnes. Ils restent dans le cadre des modèles mono-dimensionnels. Nous testons la base de test (UW) en utilisant le modèle A-R.
Les résultats concernant l'application à la base UW sont dans le tableau Tab.4.13. Ainsi les résultats (voir Tab.4.13) montrent que le modèle Traj-vertical est meilleur

	Modèle	Taux de reconnaissance
Majuscules	Traj-vertical	92.79 %
	Traj-horizontal	91.52 %
Minuscules	Traj-vertical	90.53 %
	Traj-horizontal	87.04 %
Majuscules-Minuscules	Traj-vertical	88.73 %
	Traj-horizontal	86.51 %

Tab. 4.13 – Résultats des modèles de trajectoires pour la reconnaissance des Majuscules

que le modèle Traj-horizontal du fait que les vecteurs colonnes sont discriminants pour la classification, et que les modèles de trajectoire sont meilleurs que les HMMs simples car ils prennent en considération la dépendance entre les colonnes et les lignes des observations. Cependant le temps de calcul est important puisque nous avons plus de paramètres à calculer.

4.5.4.2 Applications des modèles de trajectoire (auto-régressifs) à la base MNIST

De la même façon, nous construisons deux modèles A-R : Traj-vertical et Traj-horizontal.
Le nombre d'états optimal pour chaque classe de chiffre est celui trouvé dans le cas des HMMs simples. Ensuite nous testons nos données en utilisant ces modèles A-R. Les résultats concernant la reconnaissance sont dans le tableau Tab.4.14.

	Taux de reconnaissance
Traj-vertical	93.31 %
Traj-horizontal	91.31 %

TAB. 4.14 – Résultats de reconnaissance pour les chiffres

Ainsi ces résultats montrent que le modèle de Traj-vertical est meilleur que le Traj-horizontal, et que les modèles de trajectoires sont meilleurs par rapport aux les HMMs simples puisqu'on prend en considération les dépendances entre les observations. Par exemple le chiffre 9 de la figure Fig.4.9 est reconnu comme 4 dans le cas HMM, et comme 9 en utilisant le modèle de trajectoire.

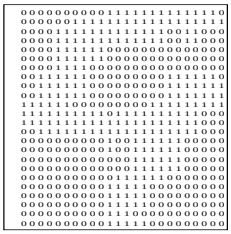


FIG. 4.9 – Chiffre 9

Jusqu'à maintenant les modèles testés dans le cadre des réseaux bayésiens sont encore des modèles mono-dimensionnels, en observant le caractère ou bien selon les colonnes ou bien selon les lignes. Notre but dans les parties suivantes, est d'utiliser les informations suivant les lignes et les colonnes conjointement. D'où l'introduction des modèles bidimensionnels.

4.5.5 Applications au modèle de fusion de données

Après avoir constaté que les HMMs verticaux sont meilleurs que les HMMs horizontaux, nous allons dans un premier temps construire un modèle qui prend en considération l'information suivant la ligne et la colonne, appelé modèle de fusion de

données similaire à celui construit sous forme HMM en utilisant HTK (multiflux). La figure Fig.4.10 représente ce modèle où les Y_t^1 (resp Y_t^2) sont les variables aléatoires dont la valeur est l'observation suivant la colonne (resp. suivant la ligne) et les X_t sont les variables d'états cachés.

Ce modèle est défini par les paramètres $A_{i,j}$, b_i^1, b_i^2 :

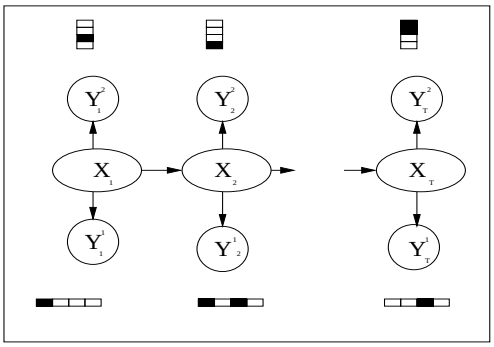


Fig. 4.10 – Modèle de fusion de données

$$\begin{cases} A_{i,j} = P(X_t \mid X_{t-1}) \\ b_i^1(.) = P(Y_t^1 = . \mid X_t = i) \\ b_i^2(.) = P(Y_t^2 = . \mid X_t = i) \end{cases} \tag{4.8}$$

Nous avons dû dans ce cas spécifique réeffectuer la recherche du nombre optimal d'états classe par classe de la même façon que dans la section 4.5.2. Le résultat de la reconnaissance est dans le tableau Tab.4.15.

base		Taux de reconnaissance
MNIST	chiffre	93.17 %
UW ENGLISH	Majuscule	93.46 %
	Minuscule	90.66 %
	Majuscules-Minuscules	89.29 %

Tab. 4.15 – Résultats de reconnaissance pour le modèle fusion de données pour les deux bases

À partir du tableau Tab.4.15, nous remarquons que le modèle de fusion de données est meilleur par rapport aux modèles HMM simples car ce modèle prend en considération l'information suivant les lignes et les colonnes. Donc c'est la première étape d'une modélisation 2D. Par contre le modèle de fusion de données a des performances sensiblement identiques au modèle de trajectoire vertical.

Les résultats sont moins bons dans le cas de l'ensemble des majuscules minuscules puisque nous avons des confusions concernant les caractères de même forme. Nous pouvons éviter ces confusions si nous employons le contexte des mots. Nous remarquons que les résultats du modèle de fusion de données sous forme HMM avec HTK sont meilleurs que ceux à base de DBN pour les mêmes raisons citées en section 4.5.2 (modèle ergodique ou gauche droite et l'initialisation). Nous avons noté l'influence de la phase d'initialisation des paramètres des modèles dans Bayesnet (dans notre cas les matrices de transitions, les moyennes et les covariances). Nous pouvons espérer avoir les mêmes résultats que dans le cas sous forme HMM si toutes les conditions décrites précédemment sont réunies.

4.5.6 Application aux modèles couplés

Afin de mieux respecter l'aspect 2D des images de caractères nous allons construire d'autres types de modèles que nous nommons modèles couplés qui sont complètement nouveaux par rapport à des modèles de type HMM. En ce qui concerne ces modèles couplés nous fixons les structures en se basant sur les critères suivants :

- le modèle construit doit avoir un nombre raisonnable de paramètres pour que la complexité du problème reste abordable.
- aucune variable continue ne doit avoir de fils discret afin de pouvoir appliquer l'algorithme d'inférence exact (JLO) ([Jensen *et al.*(1988)]).
- des liens doivent exister entre les variables cachées.

Nous rappelons que pour les modèles couplés, le choix du nombre d'états optimal est celui des HMMs correspondants. On note, pour $j = 1, 2$ $(X_t^j)_{1 \leq t \leq T}$ les variables d'états cachées et $(Y_t^1)_{1 \leq t \leq T}$ (respectivement $(Y_t^2)_{1 \leq t \leq T}$) les séquences d'observations correspondant aux colonnes des pixels du caractère (respectivement lignes) comme le montre la figure Fig.4.11.

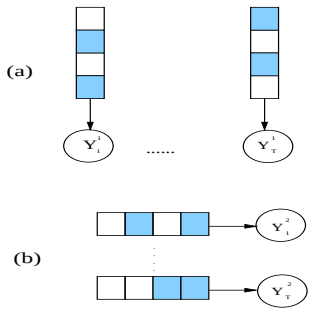


FIG. 4.11 – Séquence d'observations : (a) selon les colonnes et (b) selon les lignes

Ainsi nous construisons quatre nouveaux types de couplage à base DBN en respectant les critères décrits précédemment. Nous signalons que pour ces modèles toutes les configurations des états cachés sont autorisées (c'est une généralisation du type ergodique pour les HMMs : matrices pleines [CPDs]).

modèle-1 : ce modèle de couplage à base DBN, est le plus simple à construire, car il s'agit simplement de coupler les deux chaînes (HMMs) verticale et horizontale en liant les états cachés entre eux et en prenant en compte que le HMM vertical est meilleur que le HMM horizontal. Ceci est traduit par les directions des flèches entre les deux flux. Plus précisément les variables d'états cachées $(X_t^1)_{1\leq t\leq T}$ du flux vertical sont liées aux variables d'états $(X_t^2)_{1\leq i\leq T}$ du flux horizontal. Pour chaque $j \in \{1,2\}$ les états (X_t^j) agissent sur les observations (Y_t^j). La figure Fig.4.12 montre la structure graphique de ce modèle. Ainsi le

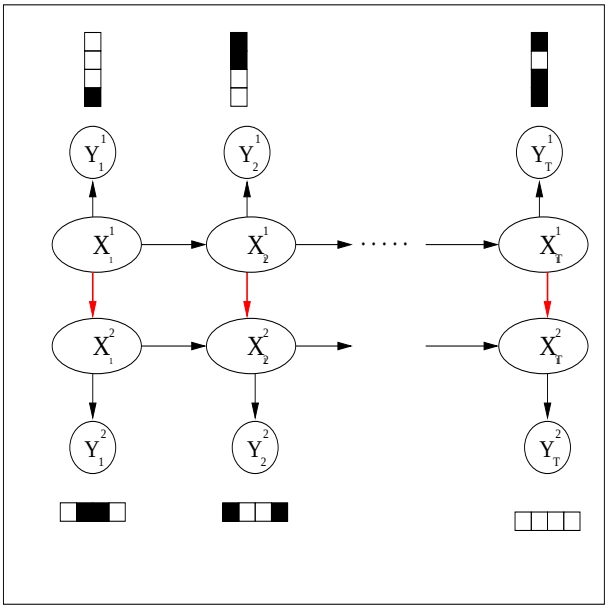


Fig. 4.12 – Premier modèle de couplage (DBN) : modèle-1

modèle est entièrement défini par la donnée des paramètres A, U, b^1 et b^2 :

$$\begin{cases} A_{i,j} = P(X_t^1 \mid X_{t-1}^1) \qquad t \geq 2 \\ U_{i,j,k} = P(X_t^2 = k \mid X_{t-1}^2 = i, X_t^1 = j) \qquad t \geq 2 \\ b_i^l(.) = P(Y_t^l = . \mid X_t^l = i) \quad \textit{pour} \quad l = 1,2 \end{cases} \tag{4.9}$$

modèle-2 : pour donner plus d'importance aux observations selon les colonnes, nous construisons un deuxième modèle. Ainsi ce modèle peut être inteprèté comme un couplage entre le modèle de fusion et le HMM vertical. Dans ce modèle à base de DBN il y a un lien supplémentaire par rapport au premier modèle c-a-d., les états (X_t^2) contrôlent les deux observations lignes et colonnes (Y_t^1) et (Y_t^2), pour $(i,j,k) \in I^3$ où I est l'espace des états. La figure Fig.4.13 montre la structure graphique de ce modèle. La paramétrisation numérique θ

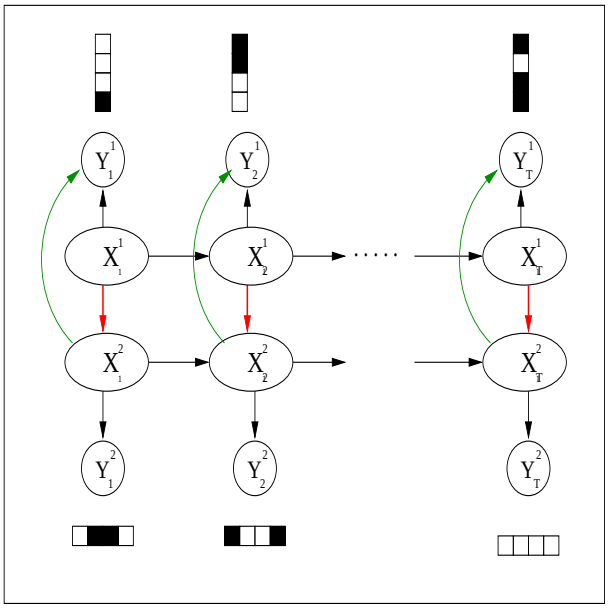


FIG. 4.13 – Deuxième modèle de couplage (DBN) : modèle-2

est définie par la donnée de A,U, b^1 et b^2 :

$$\begin{cases} A_{i,j} = P(X_t^1 \mid X_{t-1}^1) & t \geq 2 \\ U_{i,j,k} = P(X_t^2 = k \mid X_{t-1}^2 = i, X_t^1 = j) & t \geq 2 \\ b_{i,j}^1(.) = P(Y_t^1 = . \mid X_t^1 = i, X_t^2 = j) \\ b_i^2(.) = P(Y_t^2 = . \mid X_t^2 = i) \end{cases} \tag{4.10}$$

modèle-3 : pour mesurer l'influence de la dépendance des variables cachées entre elles, nous construisons un troisième modèle de couplage à base de DBN. Ce modèle diffère des précédents en ajoutant un arc dirigé de X_t^1 vers X_t^2 par rapport au premier modèle. Les liens concernant les variables observables ne sont pas changés comme le montre la fig.4.14. Ce modèle est défini par la

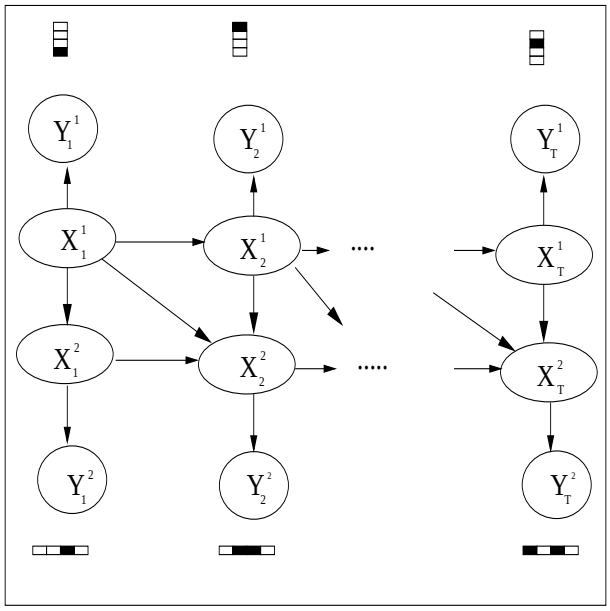


Fig. 4.14 – Troisième modèle de couplage (DBN) : modèle-3

donnée des paramètres $A_{i,j}$, $U_{i,j,k}$, b_i^1, b_i^2

$$\begin{cases} A_{i,j} = P(X_t^1 \mid X_{t-1}^1) \qquad t \geq 2 \\ U_{i,j,l,k} = P(X_t^2 = k \mid X_{t-1}^2 = i, X_t^1 = j, X_{t-1}^1 = l) \qquad t \geq 2 \\ b_i^l(.) = P(Y_t^l = . \mid X_t^l = i) \quad \textit{pour} \quad l = 1, 2 \end{cases} \tag{4.11}$$

modèle-4 : ce modèle de couplage est constitué par le couplage des deux modèles de trajectoire. Nous construisons ce quatrième modèle de couplage à base de DBN en tenant compte des dépendances entre les observations selon les lignes et les colonnes et du fait que le modèle de trajectoire vertical est meilleur que le modèle de trajectoire horizontal. Ce dernier point est traduit par les directions des flèches entre les deux flux. Dans ce modèle nous ajoutons par rapport à un couplage simple (modèle-1) des liens de dépendances entre les variables observables. La figure Fig.4.15 montre la structure de ce modèle.

Le modèle est entièrement défini par la donnée des paramètres $A_{i,j}$, $U_{i,j,k}$, B_i^1

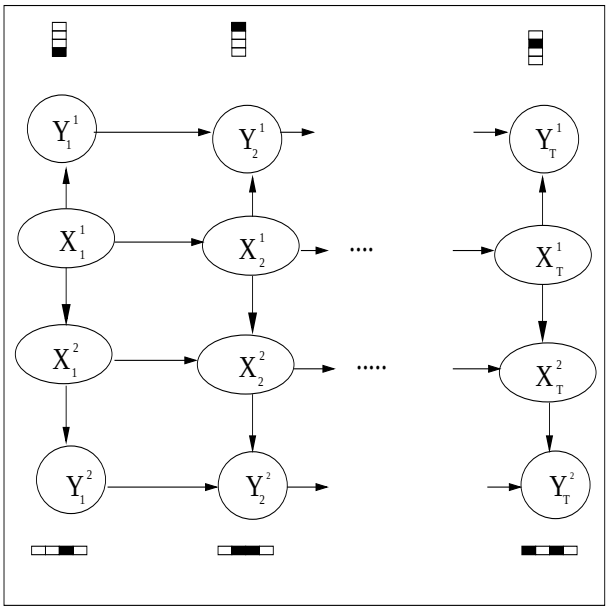


FIG. 4.15 – Quatrième modèle de couplage (DBN) : modèle-4

et B_i^2 :

$$\begin{cases} A_{i,j} = P(X_t^1 \mid X_{t-1}^1) & t \geq 2 \\ U_{i,j,k} = P(X_t^2 = k \mid X_{t-1}^2 = i, X_t^1 = j) & t \geq 2 \\ B_i^l(y_1, y_2) = P(Y_t^l = y_1 \mid Y_{t-1}^l = y_2, X_t^l = i) & l = 1, 2 \quad t \geq 2 \end{cases} \tag{4.12}$$

Contrairement à un HMM (vertical ou horizontal), ces quatre modèles sont des modèles bidimensionnels qui tiennent compte de l'information selon les lignes et les colonnes.

Dans les expériences que nous effectuons, l'apprentissage des modèles est effectué de façon indépendante classe par classe, en utilisant l'algorithme EM (Expectation Maximisation). L'étape E de l'EM est réalisée en utilisant l'algorithme d'inférence JLO [Jensen *et al.*(1988)].

Les résultats de reconnaissance pour les deux bases : UW et MNIST sont dans le tableau Tab.4.16

Base	modèle		Taux de reconnaissance
UW ENGLISH	modèle-1	Majuscules	94.78 %
		Minuscules	91.78 %
		Majuscules-Minuscules	90.76 %
	modèle-2	Majuscules	96.86 %
		Minuscules	93.33 %
		Majuscules-Minuscules	92.07 %
	modèle-3	Majuscules	95.92 %
		Minuscules	93.27 %
		Majuscules-Minuscules	90.76 %
	modèle-4	Majuscules	97.12 %
		Minuscules	93.66 %
		Majuscules-Minuscules	92.35 %
MNIST	modèle-1	Chiffres	93.79 %
	modèle-2	Chiffres	94.23 %
	modèle-3	Chiffres	93.83 %
	modèle-4	Chiffres	95.22 %

Tab. 4.16 – Résultats de reconnaissance pour les modèles couplés

À partir du tableau Tab.4.16, nous constatons que ces modèles couplés sont plus performants que les HMMs et les modèles de trajectoire non couplés pour plusieurs raisons :

- ces modèles sont bidimensionnels contrairement aux HMMs et aux modèles de trajectoires qui sont des modèles mono-dimensionnels. Donc pour ces modèles couplés nous avons plus d'information.
- nous utilisons aussi les dépendances entre les variables observables (modèle 4).
- nous favorisons les observations selon les colonnes (directions des flèches).

En comparant les modèles couplés entre eux, nous concluons que le modèle-4 concernant les couplages des modèles de trajectoires est le meilleur modèle puisque ce dernier est celui le plus proche de la réalité en faisant apparaître les dépendances entre les observations. D'autre part le modèle-2 concernant le couplage du modèle de fusion et le HMM vertical est le plus performant après le modèle-4 car il accorde plus d'importance aux observations colonnes que les deux autres modèles.
En revanche le temps de calcul est important pour tous les modèles couplés surtout pour le modèle-4 car la combinatoire des paramètres est très importante (toutes les configurations de variables d'états sont autorisées) du fait que nous utilisons une boîte à outils écrite en Matlab et un algorithme d'inférence exacte.

4.5.7 Conclusion

À partir des expériences précédentes nous pouvons déduire que :

- le HMM vertical est meilleur que le HMM horizontal du fait que les vecteurs colonnes sont plus discriminants que les vecteurs lignes pour la classification de caractères imprimés ou manuscrits.

- les résultats concernant les minuscules sont moins bons que ceux associés aux majuscules car ces derniers ont des formes plus simples.

- les taux de reconnaissance dans le cas de l'ensemble des majuscules minuscules sont moins bons que dans les cas séparés du fait de la confusion entre les caractères de mêmes formes (X et x, V et v, ...).

- les modèles de trajectoires mono-flux sont meilleurs que les HMMs simples (mono-flux). Cela est dû au fait que nous utilisons les dépendances entre variables observables dans le cas des modèles de trajectoire contrairement aux HMMs. Cependant le temps de calcul est plus important car nous sommes en présence de plus de paramètres.

- le modèle de fusion est plus performant que les deux modèles précédents du fait que ce dernier prend en considération les informations selon les lignes et les colonnes (ce modèle est plus riche au niveau des informations que les deux premiers modèles).

- les modèles couplés sont meilleurs que les modèles des HMMs simples, modèles des trajectoires et modèles de fusion. Cela est dû au fait que les modèles cou-

plés prennent en considération l'information suivant les lignes et les colonnes et aussi d'autres dépendances entre les variables (cachées ou observables) contrairement aux autres modèles. La constatation concernant les HMMs (le HMM vertical est meilleur que le HMM horizontal) est introduite dans les modèles couplés en accordant une plus grande importance aux observations colonnes qu'aux observations lignes. Ce qui est traduit par la direction des flèches des variables d'état "colonnes" vers les variables d'état "lignes" dans ces modèles couplés. En comparant les quatre modèles de couplage, et en récapitulant l'ensemble des résultats de taux de reconnaissance pour les modèles à base de DBN (voir le tableau Tab.4.17) on constate que le modèle-4 de trajectoire couplé est le meilleur modèle puisqu'il modélise bien l'image de caractère en accordant plus d'importance aux observations colonnes et en prenant en compte les dépendances entre les observations. Ce dernier résultat montre que les observations ne sont pas indépendantes contrairement à l'hypothèse "forte" généralement admise concernant les observations. Ces modèles couplés sont coûteux au niveau du temps de calcul, puisqu'il y a plus de paramètres à calculer.

- avec l'amélioration progressive de la boîte à outils Bayesnet et en résolvant le problème de l'initialisation, nous pouvons espérer améliorer encore les résultats.

Base		Modèle	Taux de reconnaissance
UW ENGLISH	Majuscules	HMM-vertical	91.35 %
		HMM-horizontal	89.67 %
		Traj-vertical	92.79 %
		Traj-horizontal	91.52 %
		fusion	93.46 %
		modèle-1	94.78 %
		modèle-2	96.86 %
		modèle-3	95.92 %
		modèle-4	97.12 %
	Minuscules	HMM-vertical	89.81 %
		HMM-horizontal	86.72 %
		Traj-vertical	90.53 %
		Traj-horizontal	87.04 %
		fusion	90.66 %
		modèle-1	91.78 %
		modèle-2	93.33 %
		modèle-3	93.27 %
		modèle-4	93.66 %
	Majuscules-Minuscules	HMM-vertical	88.21 %
		HMM-horizontal	86.13 %
		Traj-vertical	88.73 %
		Traj-horizontal	86.51 %
		fusion	89.29 %
		modèle-1	90.76 %
		modèle-2	92.07 %
		modèle-3	90.76 %
		modèle-4	92.35 %
MNIST	Chiffres	HMM-vertical	92.85 %
		HMM-horizontal	90.17 %
		Traj-vertical	93.31 %
		Traj-horizontal	91.37 %
		fusion	93.17 %
		modèle-1	93.79 %
		modèle-2	94.23 %
		modèle-3	93.83 %
		modèle-4	95.22 %

Tab. 4.17 – Résultats de reconnaissance pour les modèles à base de DBN

Chapitre 5

Conclusions et perspectives

5.1 Conclusions

Nous avons présenté dans ce travail un système de reconnaissance de caractères imprimés et manuscrits en trois parties :

- dans la première partie nous avons utilisé des classifieurs basés sur les modèles de Markov cachés gauche-droite du premier ordre dans le cas discret et le cas semi-continu. Nous avons effectué les expériences sur des ensembles distincts : majuscules, minuscules, majuscules minuscules et sur les chiffres manuscrits. Nous avons construit des modèles nommés HMM-vertical et HMM-horizontal. Les résultats de la reconnaissance montrent que d'une part les HMM-verticaux sont meilleurs par rapport aux HMMs-horizontaux. D'autre part les résultats associés aux minuscules sont moins bons que ceux associés aux majuscules du fait que les minuscules ont des formes plus complexes et nécessitent un ensemble de caractéristiques plus grand pour les décrire. Enfin nous constatons une diminution du taux de reconnaissance associé à l'ensemble majuscules minuscules car les caractères de même formes se confondent. En ce qui concerne le cas discret, les résultats sont moins bons que le cas semi-continu. Cela est dû à la perte de l'information liée à la quantification vectorielle. Cette perte d'information est liée essentiellement à la distance utilisée (SIHD) qui est invariante par translation mais sensible aux dilatations et à la construction du dictionnaire (initialisation de l'algorithme des k-moyennes).
- dans la deuxième partie nous avons remarqué que les HMMs simples (HMM-vertical et HMM-horizontal) ne prennent pas en considération les informations simultanément selon les lignes et les colonnes. Cette remarque nous a amené à construire des modèles de couplage qui respectent l'aspect 2D de l'image de caractères (ou des chiffres). Ainsi nous avons construit deux modèles de

couplage :

- modèle de fusion de données : ce modèle qui prend en considération l'information selon les lignes et les colonnes, est construit à partir d'un classifieur basé sur les HMMs.
- modèle de fusion des scores : ce modèle est construit à partir de deux classifieurs basés sur les HMMs vertical et horizontal, en pondérant les scores de décision des deux HMMs.

Les résultats obtenus montrent que les modèles de fusion sont meilleurs que les modèles HMMs simples et que le modèle de fusion de données est plus performant que le modèle de fusion des scores. Cela est dû au fait que le modèle basé sur la fusion de données prend en considération simultanément les informations selon les lignes et les colonnes tandis que le deuxième modèle basé sur la fusion des scores suppose que les classifieurs (vertical et horizontal) sont indépendants.

- dans la dernière partie, nous avons introduit les réseaux bayésiens. En effet les résultats précédents nous ont poussé à chercher de nouveaux modèles qui respectent plus l'aspect 2D de l'image des caractères. Nous avons construit à l'aide des réseaux bayésiens dynamiques quatre modèles de couplage. Ces modèles sont proches de la réalité puisqu'ils prennent en considération les dépendances entre les lignes et les colonnes de pixels des images. Pour la construction de ces modèles, on utilise l'inférence exacte (apprentissage). Les résultats obtenus montrent que les modèles de couplage basés sur les réseaux bayésiens dynamiques sont meilleurs par rapport aux autres modèles non couplés. En effet ils respectent plus l'aspect 2D de l'image des caractères et aussi la corrélation entre les lignes et les colonnes.

Nous pensons que les différences de performances constatées entre les réseaux bayésiens (outil Bayesnet) et les HMMs (outil HTK) sont dues à des aspects logiciels (initialisation) et à l'utilisation des modèles ergodiques pour les DBNs. En effet un DBN mono-flux est en théorie équivalent au HMM mono-flux correspondant (mêmes procédures de type forward-backward pour l'apprentissage). Des résultats équivalents devraient donc être atteints en améliorant la boîte à outils Bayesnet. Une fois ces conditions réalisées, on peut espérer dépasser les performances des HMMs (HTK) par les modèles de trajectoires et par les modèles couplés à base de réseaux bayésiens dynamiques dont nous avons constaté qu'ils étaient plus performants que les réseaux DBN mono-flux de type HMM.

L'apport essentiel de notre travail a consisté à montrer que les méthodes basées sur les réseaux bayésiens peuvent être appliquées à la reconnaissance des caractères

isolés (chiffres et lettres). Ensuite nous avons proposé quatre modèles de couplage respectant quelques hypothèses :

- un nombre raisonnable de paramètres pour que la complexité des calculs reste abordable.
- aucune variable continue ne doit avoir de fils discret afin de pouvoir appliquer l'algorithme d'inférence exacte (JLO).
- des liens doivent exister entre les variables cachées.

Le modèle issu du couplage des deux modèles de trajectoire (vertical et horizontal) est le plus performant, puisque ce dernier est celui qui respecte le plus les dépendances entre les observations.

5.2 Perspectives

Une poursuite envisageable de ce travail consiste à :

- Pour les HMMs :
  - utiliser une distance autre que la distance SIHD lors de la quantification vectorielle des données. Nous rappelons que la distance SIHD est invariante par translation, cependant elle est sensible aux dilatations.

  - trouver de nouvelles stratégies de fusion de scores. Notre stratégie consiste à combiner les deux scores résultants à partir des deux HMMs vertical et horizontal en favorisant les scores suivant le HMM vertical. Un super-classifieur dont les entrées seront les scores pour chaque modèle peut permettre cette fusion. Un réseau bayésien classifieur peut jouer ce rôle.

- Pour les réseaux bayésiens :
  - trouver la meilleure structure qui représente les liens (dépendances) entre les variables du réseau bayésien dynamique. Dans les applications que nous avons présentées, nous avons procédé à la fixation de la structure. Or cette stratégie de fixer la structure nous oblige à admettre quelques hypothèses fortes (par exemple : l'indépendance entre certaines variables). La recherche de la structure revient à effectuer *l'apprentissage des structures* à partir des données, qui reste un problème difficile à résoudre dans le cas d'un mélange de variables cachées.

  - déterminer les meilleures conditions initiales qui jouent un rôle important dans la phase de l'apprentissage. Pour les HMMs par exemple, l'algorithme

des k-moyennes utilisé pour l'initialisation de l'algorithme EM conduit à un apprentissage plus performant. Il s'agit donc de trouver une méthode similaire d'initialisation pour les réseaux bayésiens.

- optimiser le temps de calcul. Dans nos applications à l'aide des réseaux bayésiens, nous avons utilisé la méthode d'inférence exacte *Jonction tree*. Or cette méthode nécessite un certain nombre d'opérations que nous pouvons réduire en utilisant une méthode de calcul d'inférence approchée.

- construire des modèles couplés avec un nombre d'états cachés différent de celui trouvé pour chacune des deux chaînes (HMM ou trajectoire) verticale et horizontale. Il faudrait en fait trouver le nombre d'états optimal pour chaque classe en suivant le même protocole déjà utilisé pour les HMMs sous forme de DBN.

- étendre nos systèmes à la reconnaissance des mots. Dans ce cas il faut procéder par la segmentation des mots en caractères et effectuer la reconnaissance à partir des modèles des caractères déjà construits par nos systèmes. Pour chaque caractère la ou les meilleures hypothèses de classe peuvent être conservées à partir des vraisemblances calculées. Puis un modèle de langage peut être appliqué.

Annexe A

Adaptation du logiciel HTK pour la reconnaissance des caractères

A.1 HMMs et HTK

Les HMMs se composent d'un certain nombre d'états, de symboles observables dans chaque état, une matrice de probabilité de transition $A = (a_{ij}) = P(X_j \mid X_i)$ une matrice de probabilité d'observation $B = (b_j(K)) = P(o_k \mid X_j)$ et d'un ensemble de probabilité initiale. Dans notre application, nous avons utilisé le modèle gauche-droite :

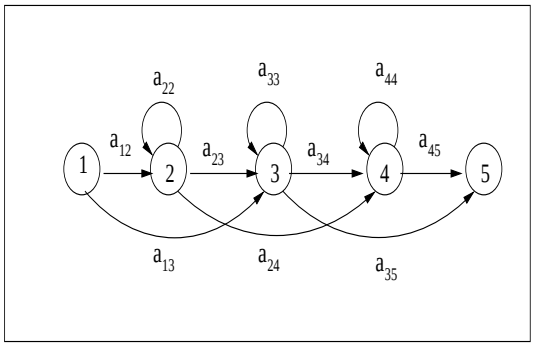


FIG. A.1 – Modèle gauche-doite

Dans le cas semi-continu la probabilité d'observation est donnée par

$$b_j(O_t) = \prod_{s=1}^{S} (\sum_{m=1}^{M_{js}} C_{jsm} \eta(o_{st}, \mu_{jsm}, \Sigma_{jsm})^{\gamma_s} \tag{A.1}$$

øù

- M_{js} est le nombre de composantes mélangées dans l'état j pour le flux ("Stream") s
- C_{jsm} est le poids de la m^{me} composante
- $\eta(., \mu, \Sigma)$ est la loi gaussienne de moyenne μ et de variance Σ telle que

$$\eta(O, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp^{-\frac{1}{2}(O-\mu)'\Sigma^{-1}(O-\mu)}$$

  n est la dimension de O.
- γ_s est le poids du stream (par défaut vaut 1)

Dans le cas discret la probabilité d'observation est donnée par :

$$b_j(o_t) = \prod_{s=1}^{S} (P_{js}(v_s(o_{st})))^{\gamma_s}$$

- $v_s(o_{st})$ est le vecteur à quantifier pour le stream s
- $P_{js}(v)$ est la probabilité de l'état j en générant le symbole v dans le stream s. Cette probabilité est donnée par :

$$P_{js}(v) = \exp(-d_{js}(v)/2371.8)$$

  avec $d_{js}(v)$ est la probabilité discrète.

A.2 Définition de base dans HTK

HTK exige une simple définition des HMMs. Pour le cas semi-continu, un exemple de fichier de définition des HMMs est donné dans la figure Fig.A.2.
Dans ce fichier, nous prenons une seule gaussienne ($M_{js} = 1$), le nombre d'états est 13 et la taille de l'observation est 50. Nous remarquons que dans tous les fichiers de définition des HMMs, le premier et le dernier état ne sont pas émetteurs.

Pour le cas discret, un exemple de fichier de définition des HMMs est donné dans la figure Fig.A.3. Dans cet exemple la taille du dictionnaire est 16 et le nombre d'états est 9.

```
~h "hmm10"
<BeginHMM>
<VecSize> 50 <USER>
  <Numstates> 13
    <State>2
     <Mean> 50
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.00.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
     <Variance> 50
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.01.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
   <State> 3
    <Mean> 50
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.00.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
      <Variance> 50
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.01.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      <State> 4
     <Mean> 50
     1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.00.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
       <Variance>50
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.01.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0


  ........

   <State>12
    <Mean> 50
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.00.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
       <Variance>50
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.01.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

   <TransP> 13
      0.0 0.6 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
      0.0 0.4 0.5 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
      0.0 0.0 0.5 0.3 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
      0.0 0.0 0.0 0.3 0.5 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0
      0.0 0.0 0.0 0.0 0.2 0.4 0.4 0.0 0.0 0.0 0.0 0.0 0.0
      0.0 0.0 0.0 0.0 0.0 0.3 0.4 0.3 0.0 0.0 0.0 0.0 0.0
      0.0 0.0 0.0 0.0 0.0 0.0 0.4 0.4 0.2 0.0 0.0 0.0 0.0
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.4 0.3 0.3 0.0 0.0 0.0
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.4 0.2 0.4 0.0 0.0
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.4 0.3 0.3 0.0
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.3 0.4 0.3
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.4 0.6
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

FIG. A.2 – Fichier de définition d'un HMM dans le cas semi-continu

```
~o<DISCRETE> <VecSize> 1 <DISCRETE>
~h "hmm5"
<BeginHMM>
  <NumStates> 9
    <State>2
      <NumMixes> 16
      <DProb> 3288*4 32767*12

      .........

      <State> 8
      <NumMixes> 16
      <DProb> 3288*4 32767*12
      <TransP> 9
       0.0 0.6 0.4 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.3 0.5 0.2 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 0.2 0.4 0.4 0.0 0.0 0.0 0.0
       0.0 0.0 0.0 0.3 0.5 0.2 0.0 0.0 0.0
       0.0 0.0 0.0 0.0 0.3 0.4 0.3 0.0 0.0
       0.0 0.0 0.0 0.0 0.0 0.3 0.4 0.3 0.0
       0.0 0.0 0.0 0.0 0.0 0.0 0.3 0.4 0.3
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

FIG. A.3 – Fichier de définition d'un HMM dans le cas discret

A.3 Réseau et dictionnaire

Avant de passer à l'étape de la reconnaissance en utilisant HTK, il faut construire un réseau qui décrit l'ordre des lettres ou des chiffres qui peuvent être identifiés et un dictionnaire qui décrit l'ordre des modèles des HMMs constituant chaque lettre (chiffre). La figure Fig.A.4 montre un exemple d'un réseau dans le cas des caractères majuscules. Dans cet exemple nous avons 29 nœuds et 53 arcs

```
VERSION=1.0
N=29  L=53
I=0   W=!NULL
I=1   W=!NULL
I=2   W=A
I=3   W=!NULL
I=4   W=B
I=5   W=C
I=6   W=D

.....

I=28  W=Z
J=0   S=0   E=2
J=1   S=0   E=4
J=2   S=0   E=5
J=3   S=0   E=6
J=4   S=0   E=7
J=5   S=0   E=8
J=6   S=0   E=9
J=7   S=0   E=10
J=8   S=0   E=11
J=9   S=0   E=12
J=10  S=0   E=13

....

J=47  S=23  E=3
J=48  S=24  E=3
J=49  S=25  E=3
J=50  S=26  E=3
J=51  S=27  E=3
J=52  S=28  E=3
```

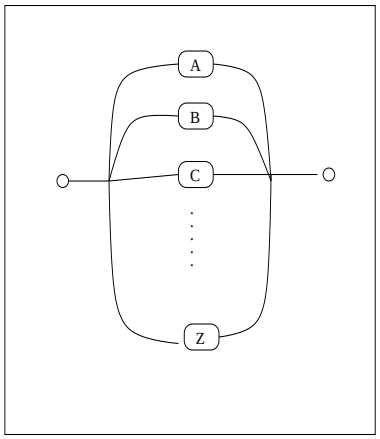


FIG. A.4 – Exemple d'un réseau

Un exemple de dictionnaire est dans la figure Fig.A.5. Le fichier hmmA correspond au modèle A .

```
A       hmmA
B       hmmB
C       hmmC
D       hmmD
E       hmmE
.....

Y       hmmY
Z       hmmZ
```

FIG. A.5 – Exemple d'un réseau

A.4 Fonctionnement de HTK

Avant de passer à la phase de la reconnaissance on construit :

- les données et les tests doivent être transformés en format htk.
- les modèles (HMMs) en utilisant les deux outils HInit et HRest.
- le dictionnaire en utilisant l'outil HDMan.
- le réseau en utilisant l'outil HParse.

Le fonctionnement global est donné dans la figure Fig.A.6

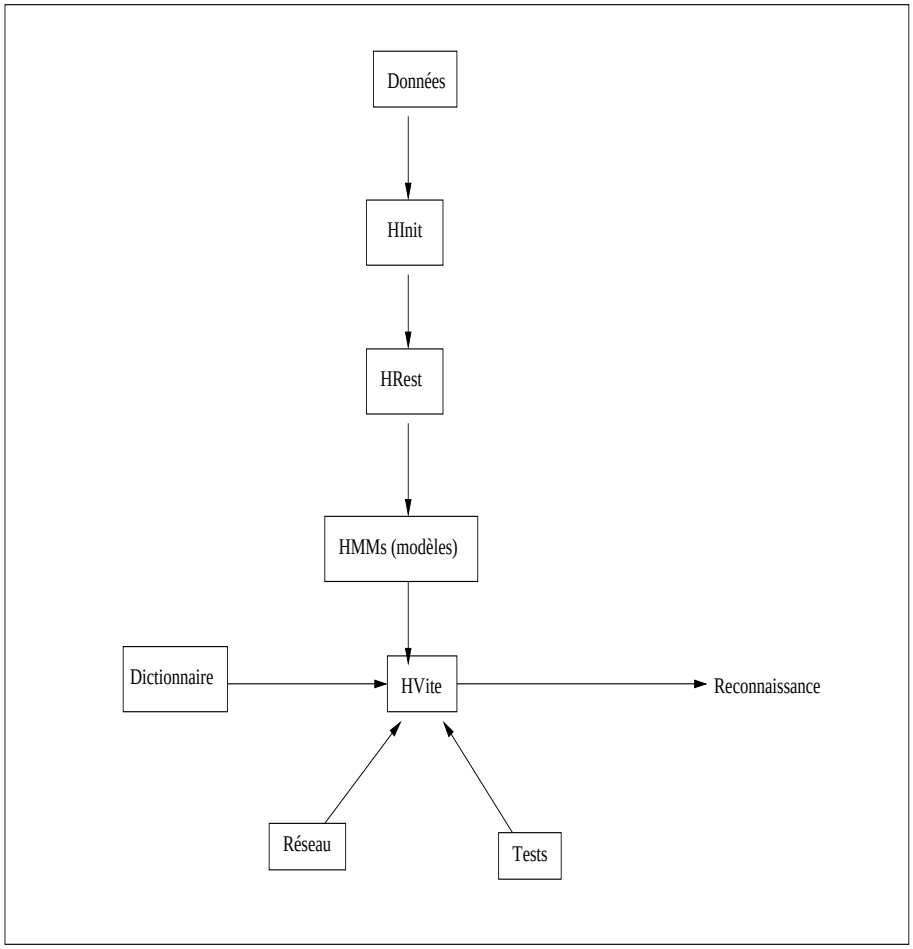


FIG. A.6 – Fonctionnement pour la reconnaissance

Annexe B

Inférence dans le cas d'un arbre

L'objectif principal des réseaux bayésiens est de calculer des probabilités conditionnelles d'événements reliés les uns aux autres par des relations cause à effet et que l'on appelle *inférence*.
Supposons que le graphe soit constitué de n nœuds notés $X = X_1, X_2, ..., X_n$. Le problème général de l'inférence consiste à calculer

$$P(X_i \mid Y) \quad \text{où} \quad Y \subset X, \quad X_i \notin Y$$

Souvent Y est l'ensemble des variables ou vecteurs d'observations noté O.
Pour calculer ces probabilités conditionnelles on peut utiliser des algorithmes d'inférence exacts ou approchés. L'algorithme le plus utilisé est l'algorithme de l'arbre de jonction [Zweig(1998)] qui est similaire à l'algorithme de Baum-Welch utilisé dans les HMMs. Pour cela on transforme le réseau initial en une nouvelle structure appelée *arbre de jonction*. Cet arbre s'obtient en suivant les étapes suivantes :

Moralisation : en reliant les parents entre eux et en éliminant les directions.

Triangularisation : en ajoutant sélectivement des arcs au graphe moral (pour ne pas avoir des cycles d'ordre quatre ou plus).

Arbre de jonction : est obtenu à partir du graphe triangulé en connectant les cliques [1] de telle façon que toutes les cliques sur le chemin entre deux cliques X et Y contiennent $X \cap Y$.

Sur un arbre, l'inférence se base sur le calcul de deux variables λ et π :

$$\lambda_j^i = P(O_i^0, O_i^- \mid X_i = j) \tag{B.1}$$

$$\pi_j^i = P(O_i^+, X_i = j) \tag{B.2}$$

où

[1] Une clique est un ensemble de variables complet maximal

O_i^0 est la valeur observée de X_i dans le cas où X_i est observable

O_i^- est l'ensemble des observations strictement en aval de X_i

O_i^+ est l'ensemble des observations strictement en amont de X_i

Ainsi on a

$$\begin{aligned} P(X_i = j, O) &= P(O_i^0, O_i^-, O_i^+, X_i = j) \\ &= P(O_i^+, X_i = j) \times P(O_i^0, O_i^- \mid X_i = j) \\ &= \lambda_j^i \times \pi_j^i \end{aligned} \tag{B.3}$$

et donc

$$P(X_i = j \mid O) = \frac{\lambda_j^i \times \pi_j^i}{\sum_j \lambda_j^i \times \pi_j^i} \quad \forall i \tag{B.4}$$

$$P(O) = \sum_j \lambda_j^i \times \pi_j^i \quad \forall i \tag{B.5}$$

Les deux variables λ et π sont analogues aux variables backward et forward α et β utilisées dans les HMMs. Ces variables peuvent se calculer de la façon suivante :

1. Calcul de λ

   si X_i est un nœud feuille alors $\lambda_j^i = 1 \ \forall j$ avec $X_i = j$

   sinon

$$\lambda_j^i = \prod_{f \in fils(Xi)} \sum_k \lambda_k^f \, P(X_f = k \mid X_i = j)$$

2. Calcul de π

   si X_i est le nœud racine alors $\pi_j^i = P(X_i = j)$

   sinon

$$\pi_j^i = \sum_v P(X_i = j \mid X_p = v) \, \pi_v^p \prod_{s \in freres(X_i)} \sum_k \lambda_k^s \, P(X_s = k \mid X_p = v)$$

   avec X_p le parent de X_i dans l'arbre

Les λ_j^i se calculent des feuilles vers la racine de l'arbre, tandis que les π_j^i se calculent de la racine vers les feuilles.

Annexe C

Algorithme EM dans les réseaux bayésiens

La méthode la plus générale de prise en compte des données incomplètes est fondée sur un algorithme de recherche itérative du modèle, inspiré de l'algorithme EM. Soit $D = (D_m)_{m=1,M}$ une base d'exemples et H un ensemble de variables cachées.
Soit

$$\begin{aligned} L &= \sum_m \log\left(P_\theta(D_m)\right) \\ &= \sum_m \log(\sum_h P_\theta(H=h, D_m)) \\ &= \sum_m \log(\sum_h q(h \mid D_m) \frac{P_\theta(H=h, D_m)}{q(h \mid D_m)} \\ &\geq \sum_m \sum_h q(h \mid D_m) \log(\frac{P_\theta(H=h, D_m)}{q(h \mid D_m)}) \qquad \text{inégalité de Jensen} \\ &= \sum_m \sum_h q(h \mid D_m) \log(P_\theta(h, D_m) - \sum_m \sum_h q(h \mid D_m) \log(q(h \mid D_m) \quad \text{(C.1)} \end{aligned}$$

où

$$q \quad \text{est une fonction telle que} \begin{cases} \sum_h q(h \mid D_m) = 1 \\ 0 \leq q(h \mid D_m) \leq 1 \end{cases}$$

- L'étape **E** (Expectation) se réalise en maximisant la limite inférieure avec l'utilisation de la fonction q. Ce qui donne :

$$q(h \mid D_m) = P_\theta(h \mid D_m)$$

- L'étape **M** (Maximization) se réalise en maximisant la limite inférieure avec l'utilisation des paramètres θ'. Cela revient à maximiser [Murphy(2002)] :

$$< l_c(\theta') >_q = \sum_m \sum_h q(h \mid D_m) \log(P(h, D_m \mid \theta')$$

Si $q(h \mid D_m) = P_\theta(h \mid D_m)$ la quantité précédente devient

$$Q(\theta', \theta) = \sum_m \sum_h P(h \mid D_m, \theta) \log(P(h, D_m \mid \theta') \tag{C.2}$$

Dempster [Dempster *et al.*(1977)] a prouvé que le choix de θ' tel que

$$Q(\theta', \theta) \geq Q(\theta, \theta) \Longrightarrow P(D \mid \theta') > P(D \mid \theta)$$

Dans le cas d'une distribution de probabilité conditionnelle multinomiale l'équation C.2 devient

$$Q(\theta', \theta) = \sum_{i,j,k} E[N_{i,j,k}] \log(\theta'_{ijk})$$

où l'espérance a posteriori courante est égale à :

$$E[N_{i,j,k}] = \sum_m P(X_i = k, C(X_i) = j \mid D_m, \theta)$$

et donc l'étape **M** : trouver θ tel que

$$\widehat{\theta} = \arg\max_{\theta'} Q(\theta', \theta) \tag{C.3}$$

devient

$$\widehat{\theta}_{i,j,k} = \frac{E[N_{i,j,k}]}{\sum_{k'} E[N_{i,j,k'}]} \tag{C.4}$$

On peut résumer l'algorithme de l'EM par :
On répète jusqu'à convergence les deux étapes *Espérance (Expectation) et Maximisation* décrites ci-dessus. Plus de détail est dans la fig C.1. L'algorithme EM possède des propriétés de convergence vers un maximum local dans certaines conditions. Dans le cas où de nombreuses données sont manquantes, il peut exister plusieurs maximum locaux. De plus EM fournit après convergence une valeur des paramètres et non une distribution pour ces paramètres. Sa simplicité en fait un des algorithmes les plus utilisés (avec des adaptations) pour l'apprentissage dans le cas de données incomplètes.

Algorithme EM

Répéter

Espérance (Expectation)
Utiliser les paramètres courants $\theta^t_{i,j,k}$ pour estimer les données manquantes

$$E^t(N_{i,j,k}) = \sum_m P^t(X_i = k, C(X_i) = j \mid D_m, \theta^t_{i,j,k})$$

Maximisation

Utiliser les données estimées pour appliquer la procédure d'apprentissage (exemple Maximum de vraisemblance)

$$\theta^{t+1}_{i,j,k} = \frac{E^t\left(N_{i,j,k}\right)}{\sum_{k'} E^t\left(N_{i,j,k'}\right)}$$

Jusqu'à convergence $(\theta^{t+1}_{i,j,k} \approx \theta^t_{i,j,k})$

Fig. C.1 – Algorithme EM pour le maximum de vraisemblance

Bibliographie

[Askilsrud *et al.*(1993)] Askilsrud, E. S., Haralick, R. M., et Phillips, I. T. (1993). *A Quick Guide to UW English Document Image Database.* version 1.0. CD-ROM, Intelligent Systems Lab, University of Washington.

[Baird(1987)] Baird, H. S. (1987). Applications of Multi-dimensional Search to Structural Feature Identification. *Workshop on Syntactical and Structural Pattern Recognition*, pages 21–24.

[Baird(1993)] Baird, H. S. (1993). *The Bell Labs Image Defect Model Database, Version 0, CD-Rom.*

[Baker(1975)] Baker, J. K. (1975). Stochastic Modeling For Automatic Speech Understanding. In *D.R. Reddy (ed.) : Speech Recognition*, pages 512–542.

[Baptista et Kulkarani(1988)] Baptista, G. et Kulkarani, K. M. (1988). A High Accuracy Algorithm for recognition of Handwritten Numerals. *Pattern Recognition*, pages 287–291.

[Baum(1972)] Baum, L. E. (1972). An inequality and Associated Maximization Technique for Probabilistic Functions of Markov Processes. *Inequalities*, **3**, 1–8.

[Baum et Egon(1967)] Baum, L. E. et Egon, J. A. (1967). An Inequality with Applications to statistical estimation for Probabilistic Functions of a Markov Process and to a model for Ecology. *Bulletin of the American Mathematical Society*, **73**, 360–363.

[Baum et Petrie(1966)] Baum, L. E. et Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov chains. *Annals of Mathematical Statistics*, **37**, 1554–1563.

[Baum et Sell(1968)] Baum, L. E. et Sell, G. R. (1968). Growth Functions for Transformations on Manifolds. *Pacific Journal of Mathematics*, **27**, 221–227.

[Baum et Sell(1970)] Baum, L. E. et Sell, G. R. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *Annals of Mathematical Statistics*, **41**, 164–171.

[Becker et Naim(1999)] Becker, A. et Naim, P. (1999). *Les Réseaux Bayésiens*. EYROLLES, première edition.

[Becker *et al.*(1996)] Becker, A., Geiger, D., et Schaffer, A. (1996). Automatic selection of loop breakers for genetic linkage analysis. *Human Heredity*, **48**(1).

[Beheim(2001)] Beheim, L. (2001). *Coopération entre segmentation et reconnaissance des caractères imprimés dégradés*. Ph.D. thesis, Université Pierre et Marie Curie.

[Belaïd et Belaïd(1992)] Belaïd, A. et Belaïd, Y. (1992). *Reconnaissance des formes : méthodes et applications*. InterEdition.

[Bellman(1957)] Bellman, J. (1957). *Dynamic Programming*. Princeton University Press.

[Bellot(2002)] Bellot, D. (2002). *Fusion de données avec des réseaux bayésiens pour la modélisation des systèmes dynamiques et son application en télémédecine*. Ph.D. thesis, Université Henri Poincaré-Nancy 1.

[Bickel et Doksum(2001)] Bickel, P. J. et Doksum, K. (2001). *Mathematical statistics*. Prentice-Hall, deuxième edition.

[Binder *et al.*(1997)] Binder, J., Koller, D., Russell, S. . J., et Kanazawa, K. (1997). Adaptative Probabilistic Networks with Hidden Variables. *Machine Learning*, pages 213–244.

[Bozinovic et Srihari(1989)] Bozinovic, R. M. et Srihari, S. N. (1989). Off-line cursive script word recognition. *IEEE Transactions on, Pattern Analysis and Machine Intelligence*, **11**, 68–83.

[Brand(1997)] Brand, M. (1997). Coupled Hidden Markov Models for modeling interacting processes. Learning and Common sense Technical Report 405.

[Chevalier *et al.*(2003)] Chevalier, S., Geoffrois, E., et Prêteux, F. (2003). A 2D dynamic programming approach for Markov random field-based handwritten character recognition. In *Proceedings of International Conference on Image and Signal Processing (ICIP)*, volume 2, pages 616–629.

[Cho et Kim(2001)] Cho, S. J. et Kim, J. H. (2001). Bayesian Network Modeling of Strokes and their Relationships for On-line Handwriting Recognition. In *Proc. of the sixth ICDAR*, pages 86–90.

[Cho et Kim(2003)] Cho, S. J. et Kim, J. H. (2003). Bayesian Network Modeling of hangul characters for on-line handwriting recognition. In *Proceeding of the Seventh International Conference on Document Analysis and Recognition (ICDAR)*, pages 207 –211.

[Choisy(2002)] Choisy, C. (2002). *Modélisation analytique de l'écriture manuscrite par une segmentation basée sur des champs de Markov.* Ph.D. thesis, Université de Nancy2.

[Choisy et Belaïd(2002)] Choisy, C. et Belaïd, A. (2002). Couplage d'une vision locale par HMM et globale par RN pour la reconnaissance de mots manuscrits. In *Conférence Internationale Francophone sur l'Écrit et le Document.*

[Chow(1965)] Chow, C. K. (1965). Statistical Independence and Threshold Functions. *IEEE Transactions of Electrical Computer*, pages 66–68.

[Cooper(1990)] Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, **42**.

[Cornuéjols et Miclet(2002)] Cornuéjols, A. et Miclet, L. (2002). *Apprentissage artificiel : concepts et algorithmes.* Eyrolles.

[Cowell et Dawid(1992)] Cowell, R. G. et Dawid, A. P. (1992). Fast retraction of evidence in a probabilistic expert system. *Statist.Comput*, **2**, 37–40.

[Cozman(1997)] Cozman, F. (1997). Robustness analysis of Bayesian networks with local convex sets of distribution. In *13th Conference on Uncertainty in Artificial Intelligence.*

[Dagum et Horvitz(1993)] Dagum, P. et Horvitz, E. (1993). A Bayesian analysis of simulation algorithms for inference in belief networks. *Networks*, **23**, 55–81.

[D'Ambrosio(1993)] D'Ambrosio, B. (1993). Incremental probabilistic inference. In *9th conference on Uncertainty in Artificial Intelligence*, pages 301–308.

[D'Ambrosio(1995)] D'Ambrosio, B. (1995). Local expression languages for probabilistic dependence. *Journal of Approximate Reasoning*, pages 61–81.

[Daoudi *et al.*(2000)] Daoudi, K., Fohr, D., et Antoine, C. (2000). A new approach for multiband speech recognition based on probabilistic graphical models. In *International conference on Spoken Language Processing (ICSLP).*

[Daoudi *et al.*(2002)] Daoudi, K., Fohr, D., et Antoine, C. (2002). Réseaux bayésiens pour la reconnaissance multi-bandes de la parole. In *XXIV ème Journées d'Étude sur la Parole.*

[Dargenton(1994)] Dargenton, P. (1994). *Contribution à la segmentation et à la reconnaissance de l'écriture manuscrite par l'ordinateur.* Ph.D. thesis, INSA de Lyon.

[Darwiche(1998)] Darwiche, A. (1998). Dynamic join trees. In *14th Conference on Uncertainty in Artificial Intelligence*, pages 97–104.

[Dawid(1979)] Dawid, A. P. (1979). Conditional independence in statistical theory. *Journal of the Royal Statistical Society*, **41**, 1–33.

[Dawid(1992)] Dawid, A. P. (1992). Applications of general propagation algorithm for probabilistic expert system. *Statistics and Computing*, **2**, 25–36.

[Dechter(1998)] Dechter, R. (1998). *Bucket elimination : a unifying framework for probabilistic inference*. MIT Press.

[Dempster *et al.*(1977)] Dempster, A. P., Laird, N. M., et Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM algorithm. *Journal of the Royal statistical Society*, **39**, 1–38.

[Devijver(1977)] Devijver, P. (1977). *Reconnaissance des formes par la méthode des plus proches voisins*. Ph.D. thesis, Université de Pierre et Marie Curie Paris VI.

[Devijver(1987)] Devijver, P. (1987). Reconnaissance des formes par la méthode des plus proches voisins. In *Workshop on Syntactical and Structural Pattern Recognition*, pages 21–24.

[Devijver et Kittler(1982)] Devijver, P. et Kittler, J. (1982). *Pattern Recognition : A statistical Approach*. Prentice Hall.

[Deviren et Daoudi(2001)] Deviren, M. et Daoudi, K. (2001). Structural learning of dynamic bayesian networks in speech recognition. In *EUROSPEECH*.

[Duda et Hart(1973)] Duda, R. et Hart, P. E. . (1973). *Pattern Classification And Scene Analysis*. JohnWiley and Sons.

[Elms(1996)] Elms, A. J. (1996). *The representation and recognition of text using hidden Markov models*. Ph.D. thesis, University of Surrey Guildford.

[Elms et Illingworth(1995)] Elms, A. J. et Illingworth, J. (1995). Modelling polyfont printed characters with HMMs and a shift invariant Hamming distance. In *ICDAR*, volume 1.

[Elms *et al.*(1998)] Elms, A. J., Procter, S., et Illingworth, J. (1998). The advantage of using an HMM based approach for faxed word recognition. *IJDAR*, **1**, 18–36.

[Finn(1996)] Finn, V. J. (1996). *An introduction to Bayesian Networks*. UCL Press, première edition.

[Forney(1973)] Forney, G. D. (1973). The Viterbi Algorithm. *Proceedings of the IEEE*, **3**, 268–278.

[Friedman et Koller(2001)] Friedman, N. et Koller, D. (2001). Being Bayesian about Network Structure : A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning*, pages 201–210.

[Friedman *et al.*(1999)] Friedman, N., Nachman, I., et Péer, D. (1999). Learning Bayesian Network Structure from Massive Datasets : The Sparse Candidate Algorithm. In *Proc. Fifteenth Conf. on Uncertainty in Artificial Intelligence (UAI)*.

[Geiger *et al.*(1990)] Geiger, D., Verma, T. S., et Pearl, J. (1990). D-separation : from theorems to algorithms. *Uncertainty in Artificial Intelligence*, pages 139–148.

[Geoffrois(2003)] Geoffrois, E. (2003). Multi-dimensional dynamic programming for statistical segmentation and recognition. In *Proceedings of International Conference on Image and Signal Processing*, volume 2, pages 397–403.

[Geoffrois *et al.*(1998)] Geoffrois, E., Jullian, A., et Debaert, C. (1998). Programmation dynamique 2d pour la reconnaissance d'images par modèles de Markov cachés. Technical report, DGA/DCE/CTA/GIP.

[Geoffrois *et al.*(2004)] Geoffrois, E., Chevalier, S., et Prêteux, F. (2004). Programmation dynamique 2D pour la reconnaissance de caractères manuscrits par champs de Markov. In *Reconnaissance des Formes et Intelligence Artificielle*.

[Ghahramani(1998)] Ghahramani, Z. (1998). Learning Dynamic Bayesian Networks. In *Adaptive Processing of Sequences and Data Structures*, pages 168–197.

[Gilloux(1994a)] Gilloux, M. (1994a). Hidden Markov models in handwriting recognition. *IMEDOVO*, pages 264–288.

[Gilloux(1994b)] Gilloux, M. (1994b). Reconnaissance de chiffres manuscrits par modèles de Markov pseudo 2D. *CNED*, pages 11–17.

[Golumbic(1980)] Golumbic, M. C. (1980). *Triangulated graphs in Algorithmic Graph Theory and Perfect Graphs*. Academic Press.

[Grandidier *et al.*(2000)] Grandidier, F., Sabourin, R., Suen, C., et Gilloux, M. (2000). Une nouvelle stratégie pour l'amélioration des jeux de primitives d'un système de reconnaissance de l'écriture. In *Colloque International Francophone sur l'Écrit et le Document (CIFED'2000)*, pages 111–120.

[Hallouli(2002)] Hallouli, K. (2002). *Utilisation de modèles markoviens pour la reconnaissance des caractères imprimés*. Rapport technique, ENST.

[Hallouli *et al.*(2002)] Hallouli, K., Likforman-Sulem, L., et Sigelle, M. (2002). A comparative study between decision fusion and fusion data in markovian printed character recognition. In *Actes ICPR*, Quebec.

[Hallouli *et al.*(2003)] Hallouli, K., Likforman-Sulem, L., et Sigelle, M. (2003). Réseaux bayésiens dynamiques pour la reconnaissance des caractères imprimés dégradés. In *GRETSI*, Paris.

[Hamilton(1990)] Hamilton, J. (1990). Analysis of time series subject to changes in regime. *J.Econometrics*, **45**, 39–70.

[Hamilton(1994)] Hamilton, J. (1994). *Time Series Analysis.* Wiley.

[Heckerman *et al.*(1992)] Heckerman, D., Horvitz, E., et Nathawani, B. (1992). Towards normative expert systems : Part I. *Methods of information in Medecine*, **31**, 90–105.

[Heckerman *et al.*(1994)] Heckerman, D., Geiger, D., et Chickering, D. (1994). Learning Bayesian networks : The Combination of Knowledge and Statistical Data. *Machine Learning*, **20**, 197–243.

[Hildebrandt et Liu(1993)] Hildebrandt, T. et Liu, W. (1993). Optical recognition of handwritten Chinese Characters : advances since 1980. In *Pattern recognition*, pages 205–225.

[Huang et Darwiche(1996)] Huang, C. et Darwiche, A. (1996). Inference in Belief Networks : A procedural guide. *Intl. Journal of Approximate Reasoning.*, **15**, 225–263.

[Jaakkola et Jordan(1999)] Jaakkola, J. T. S. et Jordan, M. I. (1999). Variational probabilistic inference and the QMR-DT network. *Journal of Artificial Intelligence Research*, **10**, 291–322.

[Jelinek(1997)] Jelinek, F. (1997). *Statistical methods for speech recognition.* MIT Press.

[Jensen et Jensen(1994)] Jensen, F. et Jensen, V. (1994). Optimal junction trees. In *Proceeding of the 10th conference on Uncertainty in Artifical Intelligence*, pages 360–366.

[Jensen(1995)] Jensen, F. V. (1995). Cautious propagation in bayesian networks. In *Proceeding of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 323–328.

[Jensen(1996)] Jensen, F. V. (1996). *An introduction to bayesian networks.* UCL Press.

[Jensen *et al.*(1988)] Jensen, F. V., Lauritzen, S. L., et Olesen, K. (1988). Bayesian updating in causal probabilistic networks by local computations. *Comp.Stat.Quart*, **4**, 269–282.

[Jensen(2001)] Jensen, V. F. (2001). *Bayesian Networks and Decision Graphs.* Springer, 175 Fifth Avenue, New York NY 10010 USA, première edition.

[Jordan(1998)] Jordan, M. I. (1998). *Learning in Graphical Models.* MIT Press, Five Cambridge Center, MA 02142-1493 USA, première edition.

[Jordan(2001)] Jordan, M. I. (2001). *Graphical Models*. MIT Press, Five Cambridge Center, MA 02142-1493 USA, première edition.

[Jordan et Weiss(2001)] Jordan, M. I. et Weiss, Y. (2001). Probabilistic inference. In *Graphical models*. MIT Press, Five Cambridge Center, MA 02142-1493 USA.

[Jordan *et al.*(1999)] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., et Saul, L. K. (1999). An introduction to variational methods for graphical models. In *Machine Learning*, volume 37, pages 183–233. Cambridge : MIT Press.

[Kharroubi(2002)] Kharroubi, J. (2002). *Etudes de techniques de classement « Machines à vecteurs supports » pour la vérification automatique du locuteur*. Ph.D. thesis, ENST.

[Kittler *et al.*(1998)] Kittler, J., Hatef, M., Duin, R. P. W., et J.Matas (1998). On Combining Classifiers . *IEEE Transactions on Pattern Analysis and Machine intelligence*, **20**, 226–239.

[Kjaerulf(1990)] Kjaerulf, U. (1990). *Triangulation of graphs-algorithms giving small total state space*. Dept. of Math. and Comp.Sci, technical report r-90-09 edition.

[Kordi *et al.*(1987)] Kordi, K. ., Xydeas, C., et Holt, M. (1987). Printed Character Recognition Using Markov Models. In *Onzième Colloque Gretsi*, pages 507–509.

[Lauritzen et Spiegelhalter(1988)] Lauritzen, L. S. et Spiegelhalter, J. D. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Royal Statistical Society*, pages 157–282.

[Lauritzen *et al.*(1999)] Lauritzen, L. S., Spiegelhalter, J., Cowell, G. R., et Dawid, P. A. (1999). *Probabilistic Networks and Expert Systems*. Springer, 175 Fifth Avenue, New York NY 10010 USA, première edition.

[Lauritzen(1996)] Lauritzen, S. (1996). *Graphical Models*. Clarendon Press.

[Lauritzen *et al.*(1990)] Lauritzen, S. L., Dawid, A. P., Larsen, B. N., et Leimer, H. G. (1990). Independence properties of directed Markov fields. *Networks*, **50**, 491–505.

[Lebourgeois(1991)] Lebourgeois, F. (1991). *Approche mixte pour la reconnaissance des documents imprimés*. Ph.D. thesis, Institut National des Sciences Appliquées de Lyon.

[LeCun(1998)] LeCun, Y. (1998). The MNIST handwritten digit database. Available on the web at http :// www.research.att.com/ yann/ocr/mnist/.

[LeCun *et al.*(2001)] LeCun, Y., Bottou, L., Bengio, Y., et Haffner, P. (2001). Gradient-Based Learning Applied to Document Recognition. In *Intelligent Signal Processing*, pages 306–351.

[Lin et Druzdzel(1998)] Lin, Y. et Druzdzel, M. J. (1998). Relevant -based sequential evidence processing in Bayesian networks. In *11th International FLAIRS Conference*, pages 446–450.

[Mackay(1999)] Mackay, D. (1999). An introduction to Monte Carlo methods. In *Learning in Graphical Models*, pages 175– 204. MIT Press.

[Madsen et D'Ambrosio(1999)] Madsen, A. L. et D'Ambrosio, B. (1999). Lazy Propagation and Independence of Causal Influence. In *5th European Conference on symbolic and Quantative Approaches to reasoning with Uncertainty*, pages 293–304.

[Madsen et Jensen(1999)] Madsen, A. L. et Jensen, F. V. (1999). LAZY propagation : A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, **5**, 203–245.

[Meek *et al.*(2002)] Meek, C., Chickering, D. M., et Heckerman, D. (2002). Autoregressive tree models for time-series analysis. In *Proceedings of the second International SIAM Conference on DataMining*, pages 229–244.

[Miclet(1984)] Miclet, L. (1984). *Méthodes structurelles pour la Reconnaissance des Formes*. Eyrolles.

[Milgram(1993)] Milgram, M. (1993). *Reconnaissance des formes, méthodes numériques et connexionnistes*. A.Colin.

[Milgram et Schwenk(1996)] Milgram, M. et Schwenk, H. (1996). Reconnaissance de Codes Postaux par Réseaux Diabolos. *Colloque National sur l'Ecrit et le Document (CNED 96)*.

[Murphy(2001)] Murphy, K. (2001). The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, **33**.

[Murphy(2002)] Murphy, K. (2002). *Dynamic Bayesian Networks : Representation, Inference and Learning*. Ph.D. thesis, University of California, Berkeley.

[Murphy(2003)] Murphy, K. (2003). Bayes net toolbox for matlab. Available on the web at http ://www.ai.mit.edu/murphyk/Bayes/bnintro.html, October 2003.

[Murphy et Mian(1999)] Murphy, K. et Mian, S. (1999). Modelling Gene Expression Data using Dynamic Bayesian Networks. Technical report, UC Berkeley, CA.

[Park et Lee(1998)] Park, H. et Lee, S. (1998). A truly 2-D Hidden Markov Model for Off-Line Handwritten Character Recognition. *Pattern Recognition*, **31**, 1849–1864.

[Pearl(1982)] Pearl, J. (1982). Reverend Bayes on Inference Engines : Distributed Hierarchical Approach. In *Proceedings, AAAI National Conference on Artificial Intelligence*, pages 133–136.

[Pearl(1988)] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufman, deuxième edition.

[Rabiner(1988)] Rabiner, L. R. (1988). Mathematical Foundations of Hidden Markov Models. *Recent Advances in speech Understanding and Dialog Systems*, **46**, 183–205.

[Rabiner(1989)] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2), 257–286.

[Rabiner et Juang(1986)] Rabiner, L. R. et Juang, B. H. (1986). An introduction to Hidden Markov Models. *Recent Advances in speech Understanding and Dialog Systems*, **3**, 4–16.

[Ramdane *et al.*(2002)] Ramdane, S., Taconet, B., Zahour, A., et Faure, A. (2002). Apprentissage non supervisé des modèles de Markov cachés. In *Colloque International Francophone sur l'Écrit et le Document (CIFED'2002)*, pages 61–70.

[Ramesh(1989)] Ramesh, S. R. (1989). A generalized character recognition algorithm : A graphical approach. *Pattern Recognition*, **22**, 347–350.

[Recht(1999)] Recht, M. (1999). Contribution à la reconnaissance de caractères dégradés par chaînes de Markov cachées (HMM). Mèmoire de DEA ENSEA, Cergy-Pontoise.

[Russell et Norvig(2002)] Russell, S. et Norvig, P. (2002). *Artificial Intelligence : A Modern Approach*. Prentice Hall, deuxième edition.

[Sang *et al.*(2003)] Sang, L., Wu, Z., Yang, Y., et W.Zhang (2003). Automatic speaker recognition using dynamic bayesian network. In *ICASSP'03*.

[Saon(1997)] Saon, G. (1997). *Modèles Markoviens uni et bidimensionnels pour la reconnaissance de l'écriture manuscrite hors-ligne*. Ph.D. thesis, Université Henri Poincaré - Nancy1.

[Saul et Jordan(1995)] Saul, L. et Jordan, M. (1995). Boltzmann chains and Hidden Markov Models. In *NIPS-7*.

[Schmidt et Shenoy(1998)] Schmidt, T. et Shenoy, P. P. (1998). Some improvements to the Shenoy-Shafer and Hugin architectures for computing marginals. *Artificial Intelligence*, pages 323–333.

[Shafer et Shenoy(1990)] Shafer, G. R. et Shenoy, P. P. (1990). Probability propagation. *Annals of Mathematics and Artificial Intelligence*, **2**, 327–352.

[Shenoy et Shafer(1990)] Shenoy, P. et Shafer, G. (1990). Axioms for probability and belief-function propagation in Uncertainty and Artificial Intelligence. In *Uncertainty in Artif. Intell*, volume 4, pages 169–198.

[Shenoy(1997)] Shenoy, P. P. (1997). Binary join trees for computing marginals in the Shenoy-Shafer architecture. *Internat.J.Approx.Reason.*, pages 239–263.

[Sherkat et Allen(1999)] Sherkat, N. et Allen, T. J. (1999). Whole Word Recognition in Facsimile Images. In *Fifth International Conference on Document Analysis and Recognition*, pages 547–550.

[Simard *et al.*(2003)] Simard, P. Y., Steinkraus, D., et Plat, J. C. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis . In *Seventh International Conference on Document Analysis and Recognition Volume II*, pages 958–962.

[Souafi(2002)] Souafi, S. (2002). *Contribution à la reconnaissance des structures des documents écrits : Approche probabiliste.* Ph.D. thesis, INSA de LYON.

[Tou et C.Gonzalez(1974)] Tou, J. T. et C.Gonzalez, R. (1974). *Pattern Recognition Principles.* Addison-Wesley.

[Vapnik(1998)] Vapnik, V. (1998). *Statistical learning theory.* J. Willey.

[Vinciarelli et Luettin(2000)] Vinciarelli, A. et Luettin, J. (2000). Off-Line Cursive Script Recognition Based on Continuous Density HMM. In *International Workshop on Frontiers in Handwriting Recognition, IWFHR'2000*, pages 493–498.

[Williams *et al.*(2000)] Williams, W. J., Zalubas, E. J., et Hero, A. O. . (2000). Word Spotting in Bitmapped Fax Documents. *Information Retrieval*, **2**, 207–226.

[Xiang *et al.*(2000)] Xiang, Y., Olesen, K. G., et F.V.Jensen (2000). Practical Issues in Modeling Large Diagnostic Systems with Multiply Sectioned Bayesian Networks. *International Journal of pattern Recognition and Artificial Intelligence*, **14**(1), 59–72.

[Xiao et Leedham(2002)] Xiao, X. et Leedham, G. (2002). Signature verification using a modified Bayesian network. *Pattern Recognition*, **35**, 983–995.

[Xu(1994)] Xu, H. (1994). Computing marginals from the marginal representation in Markov trees. In *International Conference on Information processing and Management of Uncertainty in knowlege based systems (IPMU)*, pages 275–280.

[Young(2001)] Young, S. J. (2001). HTK : Hidden Markov Model Toolkit V3.0. Available on the web at http ://htk.eng.cam.ac.uk/.

[Zhang et Poole(1994)] Zhang, N. L. et Poole, D. (1994). Intercausal independence and heterogeneous factorization. In *10th Conference on Uncertainty in Artificial Intelligence*, pages 606–614.

[Zhang et Poole(1996)] Zhang, N. L. et Poole, D. (1996). Exploiting causal independence in Bayesian networks inference. *Journal of Artificial Intelligence Res*, **5**, 301–328.

[Zhang et Yan(1997)] Zhang, N. L. et Yan, L. (1997). Independence of Causal Influence and Clique Tree Propagation. *International Journal of Approximate Reasoning*, **19**, 335–349.

[Zweig(1998)] Zweig, G. (1998). *Speech Recognition with Dynamic Bayesian networks*. Ph.D. thesis, University of California, Berkeley.