Rapport de stage de DEA

François Olivier

Mai à septembre 2002

Résumé

Le formalisme des réseaux bayésiens est une alternative efficace à la gestion de connaissance, car ils sont le résultat d'une convergence entre les méthodes statistiques, qui permettent le passage de l'observation à la loi, et des technologies de l'intelligence artificielle, qui permettent aux ordinateurs de traiter de la connaissance plutôt que de l'information. En effet, les chercheurs se sont rapidement aperçus que la construction d'un système expert nécessitait presque toujours la prise en compte de l'incertitude dans le raisonnement.

L'objectif de ce stage, effectué auprès de M. Leray Philippe, Maître de Conférences à l'I.N.S.A. de Rouen, laboratoire PSI, est multiple :

- D'une part il est la suite logique du stage qui s'est déroulé fin 2001 à l'I.N.S.A. et qui consiste en la mise en œuvre d'un réseau bayésien pour faire de l'aide au diagnostic médical pour la détection du cancer de la thyroïde.
- Il aboutit en différentes implémentations Matlab utilisant la "Bayes Net Toolbox" écrite par K. Murphy et en une mise à jour du site français de la BNT (http ://bnt.insa-rouen.fr/)
- D'autre part, il consiste en une recherche bibliographique sur les différentes facettes du sujet, surtout sur les problèmes liés à l'apprentissage de la structure, ainsi que sur les différentes applications des réseaux bayésiens. Il a débouché sur la participation à un article scientifique [LER02] soumis à la Revue d'Intelligence Artificielle.

Ce travail est séparé en quatre parties :

1°) Généralités des réseaux bayésiens,

2°) Apprentissage des paramètres,

3°) Les différentes méthodes d'apprentissage de la structure, illustrées par un exemple,

4°) Implémentation et résultats obtenus à l'aide de la base de données *Thyroid*.

Mots-clés : réseaux bayésiens, diagnostic médical, inférence bayésienne, apprentissage des paramètres à partir de données, apprentissage de la structure, algorithmes de causalité, algorithmes de score, équivalents de Markov, bayes net toolbox.

Liste des tableaux

Table des figures

1.1 Introduction

Depuis l'origine de la recherche scientifique sur l'incertain (Pascal 1623-1662, Bernouilli 1700-1782, Laplace 1749-1787), la théorie des probabilités n'était prévue que pour s'appliquer sur des expériences indéfiniment répétables. Or grâce au travaux de Kolmogorov 1903-1987, De Finetti 1906-1985, Ramsey 1903-1930, une nouvelle notion est apparue : les probabilités subjectives. S'appuyant sur les travaux de Bayes 1702-1761, elle a permis d'étendre le cadre d'application des probabilités au cas où l'expérience que l'on veut simuler ne peut pas être itérée plusieurs fois au préalable. Puis grâce aux travaux de Von Neumann 1903-1957 et Morgenstern 1902-1976 la théorie de l'espérance d'utilité est née, et elle a permis la création d'un domaine plus vaste : la théorie de la décision.

Celle-ci permet cette fois d'évaluer les risques et gains probables sur une expérience que l'on ne va effectuer qu'une seule fois et pour laquelle nos décisions, tout au long de cette expérience peuvent être fatales (en médecine particulièrement) ou même ruineuses (en logistique d'entreprise ou en spéculation boursière par exemple).

Les réseaux bayésiens sont des outils généraux développés récemment [KIM87], [LAU88], [JEN96], [JOR98]. Ils sont un mode intéressant de représentation d'une structure d'influence entre divers faits, états ou hypothèses d'états. Ils permettent de modéliser des informations telles que des dépendances causales, spatiales, temporelles, même si elles sont imparfaites ou manquantes.

Par exemple, *"en médecine, une même combinaison de symptômes peut-être observée dans différentes pathologies. Il n'y a donc pas de règles strictes qui permettent de passer systématiquement d'un ensemble d'observations à un diagnostic. De plus, les informations pertinentes ne sont pas toujours observables. Pour que des systèmes experts puissent être utilisés dans de tels domaines, il faut donc qu'ils soient capables de raisonner sur des faits et des règles incertains. Cependant l'approche des réseaux bayésiens est essentiellement différente de celle de la logique floue : les réseaux bayésiens s'attachent à la prise en compte de faits précis, mais incertains, alors que la logique floue s'intéresse à la modélisation de faits imprécis."* (extrait de [NAI99])

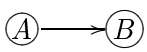
Le formalisme des réseaux bayésiens est donc l'un des rares outils scientifiques qui servent à la fois à prévoir et à détecter, et cela, même lorsque la situation est incertaine et les données incomplètes. Il est donc particulièrement adapté à la détection de fraude (alarmes), au diagnostic médical, aux études de marché, à l'évaluation de risque (spéculation boursière), à l'intelligence artificielle...

1.2 Les réseaux bayésiens

1.2.1 Généralités

A partir d'un ensemble d'informations (que nous appellerons données), on doit pouvoir extraire la connaissance noyée dans ces chiffres, qui nous servira à raisonner grâce à la structure du réseau, et en déduire le ou les résultats possibles avec une mesure qualitative de la pertinence de cette réponse.

Or les réseaux bayésiens sont à mi-chemin entre les probabilités (fréquentistes et subjectives) et les modèles graphiques, car la manière la plus intuitive pour représenter un lien de cause à effet est simplement une flèche.



Ce qui signifie que la connaissance que l'on a sur A a une influence sur celle que l'on a sur B. Mais il est important de voir que ceci est réciproque, et la connaissance ou l'absence

veuille alors calculer la valeur de la loi de probabilité de B, ici un tel réseau représente une loi jointe $\mathbb{P}(A,B) = \mathbb{P}(A).\mathbb{P}(B/A)$ ce qui est la formule de Bayes, et la flèche signifie que à priori on pense que $P(B/A) \neq P(B)$ (sinon cela signifie que les deux variables sont indépendantes ($\mathbb{P}(A,B) = \mathbb{P}(A).\mathbb{P}(B)$) et il n'y a donc pas de raison de les relier).

Quand on reçoit de l'information sur $\mathbb{P}(A)$ et que l'on veut mettre à jour la loi $\mathbb{P}(B)$, il suffit juste que le nœud A envoie sa loi au nœud B et que celui-ci effectue le calcul $\mathbb{P}(B) = \mathbb{P}(B/A).\mathbb{P}(A)$.

Les premiers réseaux bayésiens étudiés sont les chaînes de Markov, et l'exemple précédent n'est qu'une chaîne de longueur 2, et l'on voit bien que pour une chaîne de Markov de longueur n, la loi jointe, donnée par la formule de Bayes,

$$\mathbb{P}(x_1, \cdots, x_n) = \mathbb{P}(x_1).\mathbb{P}(x_2/x_1).\mathbb{P}(x_3/x_1, x_2) \cdots \mathbb{P}(x_n/x_1, \cdots, x_{n-1}), \tag{1}$$

va jouer un rôle central et se simplifier en

$$\mathbb{P}(x_1, \cdots, x_n) = \mathbb{P}(x_1).\mathbb{P}(x_2/x_1).\mathbb{P}(x_3/x_2) \cdots \mathbb{P}(x_n/x_{n-1}).$$

Définition 1 *Un réseau bayésien est constitué de la réunion d'un* graphe acyclique dirigé (D.A.G.) $G = (X, U)$ *dont les sommets représentent un ensemble de variables aléatoires* $X = \{X_1, \cdots, X_n\}$ *et de matrices de probabilités conditionnelles* $[\mathbb{P}(x_i/x_{pa(X_i)})]$ *où* $pa(X_i)$ *représente l'ensemble des indices des parents de* X_i *dans le graphe (avec* $x_\emptyset = \emptyset$ *et* $x_{\{a\}\cup E}$ signifie $X_a = x_a$ *et* x_E*).*

Et ceci permet de déduire une expression de la loi jointe (1) en fonction de la structure du graphe :

$$\mathbb{P}(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} \mathbb{P}(x_i/x_{pa(X_i)}) \tag{2}$$

1.2.2 La d-séparation

On va pouvoir déduire s'il y a ou pas indépendance conditionnelle pour tous les groupes de variables, et non pas seulement à ceux présents dans la décomposition de la loi jointe, grâce au critère suivant :

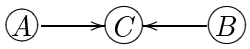
Définition 2 (chaîne active) *Soit X, Y, et Z trois ensembles disjoints de variables. On dit qu'une chaîne entre X et Y est active par rapport à Z, si les deux conditions suivantes sont réunies :*

- *tout sommet à arcs convergents des chaînes qui relient un sommet de X à un sommet de Y est dans Z ou a un descendant dans Z,*
- *aucun autre sommet de la chaîne n'est dans Z.*

Une chaîne qui n'est pas active est dite bloquée par Z.

Définition 3 (d-séparation) *On dit que Z d-sépare X et Y, et on note* $X|d|Y/Z$*, si toute chaîne reliant X à Y est bloquée par Z.*

Définition 4 *On appelle* V-structure de sommet puits C*, tout sous-graphe de la forme suivante :*



En clair, si on trouve **une** V-structure avec son sommet *puits* qui est, ou un de ses descendants, dans l'ensemble de conditionnement, alors il **n**'y a **pas** d-séparation (voir figure 1). L'intérêt d'une telle notion réside dans le théorème suivant :

Théorème 1.1 (Verma et Pearl, 1988) *Soit G un graphe d'indépendance, alors pour tous X, Y et Z sous-ensembles disjoints de variables,*

$$X|d|Y/Z \iff \text{la décomposition dont G est issu entraîne que } X \perp Y/Z \tag{3}$$

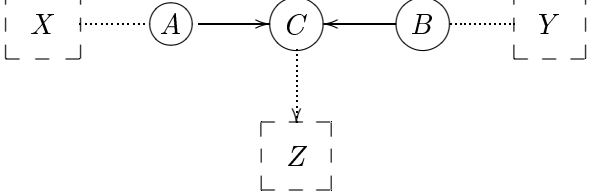


FIG. 1 – Illustration d'un nœud *puits : C*.

De plus on peut remarquer que la séparation au sens classique (lorsque le retrait d'un nœud coupe le graphe en deux parties) implique la d-séparation.

Dans le cas d'une arborescence, la d-séparation est équivalente à la séparation, car il n'y a qu'un seul parent.

1.3 L'inférence bayésienne

Par la suite on appelera D.A.G, un graphe acyclique dirigé.

Il existe deux algorithmes pour les calculs exacts dans les réseaux bayésiens. La première méthode, qui ne nécéssite aucun traitment du graphe, est plus intuitive. Seulement cette méthode ne marche que pour les D.A.G sans cycle. La seconde, consiste à effectuer des transformations sur le graphe pour obtenir un arbre.

1.3.1 Les messages locaux

L'inférence dans les réseaux bayésiens par envois de messages locaux a été introduite par KIM et PEARL en 1985 (cf [KIM87]). Elle consiste en une actualisation, à tout moment, des probabilités marginales, par transmission de messages entre variables voisines dans le graphe d'indépendence. Cette méthode ne fonctionne de manière exacte que lorsque le réseau bayésien est un *arbre*, cependant une version approchée existe lorsque l'on est en présence d'un D.A.G quelconque.

⁕ Cette méthode a été implémentée sur BNT sous la fonction **global_joint_inf_engine** (trois versions spécialisées ont également été écrites : **enumerative_inf_engine**, **gaussian_inf_engine** et **cond_gauss_inf_engine**).

1.3.2 L'arbre de jonction

Cette méthode a été introduite par Lauritzen, Spiegelhalter [LAU88] et Jensen [JEN90], elle est applicable sur tout D.A.G, contrairement à la méthode précédente.

Elle consiste en une transformation du graphe d'origine en graphe moral triangulé, puis en arbre de jonction.

Moralisation : La première étape consiste à *'marier les parents'*, c'est-à-dire que lorsqu'un nœud a plusieurs parents, on les relie tous 2 par 2 par une arête, et on obtient le graphe dit *moral*.

Le fait d'avoir retiré les orientations, on pourrait croire que l'on a perdu de l'information, mais cela n'est pas le cas.

à des sous-graphes complets, il suffit de procéder à la triangulation du graphe moral en y ajoutant des arêtes créant des raccourcis dans tout cycle de longueur 4 ou plus, on obtient alors un graphe moral *triangulé* (qui n'est pas unique).

L'arbre de jonction : On va pouvoir construire un arbre dans lequel il sera possible de faire de l'inférence grâce aux résultats suivants :

Propriété 1.2 (de l'intersection courante) *Pour toute clique $\mathcal{C}_i$ il y a une clique $\mathcal{C}_{j(i)}$ avec $j(i) < i$ telle que toute variable commune à $\mathcal{C}_i$ et $\mathcal{C}_{i'}$ où $i' < i$ appartient à $\mathcal{C}_{j(i)}$.*

Définition 5 *Un* arbre de jonction *est un arbre ayant l'ensemble des cliques d'un graphe triangulé comme ensemble de nœuds et vérifiant la propriété de l'intersection courante.*

Théorème 1.3 *A partir d'un graphe moral triangulé on peut toujours construire un arbre de jonction.*

Les variables de l'arbre de jonction sont donc des groupes de variables du graphe d'origine. Par envois de messages (les potentiels de cliques) dans l'arbre de jonction on peut faire une inférence dans n'importe quel D.A.G. On va donc mettre à jour les potentiels de cliques, à partir desquels on pourra alors déduire les lois marginales pour nos variables d'origines.

⁎ Cette méthode est implémentée sous BNT dans la fonction **jtree_inf_engine**.

1.3.3 Les méthodes approchées

Les méthodes statistiques ne calculent pas exactement les lois marginales, mais en donne un estimation. Elles ont l'avantage de converger rapidement et permettent donc de traiter les applications de grande taille, ce qui n'est pas le cas des méthodes exactes.

De nombreux algorithmes ont été développés, citons par exemple les méthodes MCMC (Monte-carlo Markov Chains, [GIL96]), l'échantillonnage de Gibbs [LAU96], l'algorithme de Metropolis-Hastings et les méthodes variationnelles.

1.3.4 *Most probable explanation*

Lorsque l'on ne s'intéresse pas à connaître les probabilités à postériori, mais seulement quel est le cas le plus probable, il est possible de faire une inférence nettement plus rapide, en utilisant des "*max*" à la place des sommes et des "*min*" à la place des produits dans la propagation, et même si les valeurs de probabilités données par cette méthode sont fausses, le cas le plus probable reste celui que l'on trouve par cette méthode. On peut même adapter cette méthode pour trouver le deuxième cas le plus probable, le n-ième cas le plus probable.

⁎ Cette fonction existe dans la Bayes Net Toolbox par le biais de la fonction **calc_mpe**.

1.4 Des réseaux remarquables

1.4.1 Le réseau naïf

En diagnostic médical, comme dans de nombreux domaines, il est souvent difficile de connaître la structure du réseau, car celui-ci est censé représenter toutes les variables intervenantes, ainsi que leurs liens causaux, or on ne connaît parfaitement ni l'un ni l'autre. Pour éviter ces problèmes on peut construire un réseau dit naïf. L'intuition nous donnerait le réseau suivant :

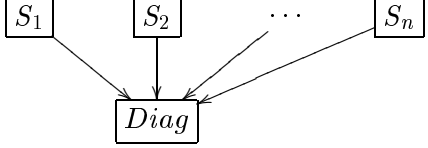


FIG. 2 – le réseau intuitif.

Or on a tendance à inverser toutes les flèches, ce qui revient à ajouter l'hypothèse d'indépendance conditionnelle des symptômes sachant le diagnostic, car toutes les V-structures disparaissent.

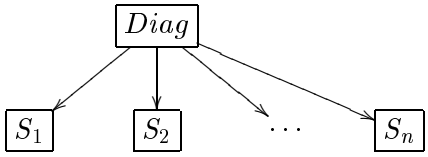


FIG. 3 – le réseau naïf.

Cette manipulation a en plus l'avantage suivant : pour le graphe de la figure 2, la matrice de probabilités conditionnelles stockée au nœud diagnostic est de taille $2^{nombre\ de\ symptômes}$ (cf définition 1 et en supposant que tous les nœuds sont binaires), ce qui peut vite être énorme. Alors que pour le graphe 3, on a juste une matrice de taille 4 (toujours dans le cas binaire) à stocker en chaque nœud symptôme, et plusieurs petites matrices sont plus simples à gérer qu'une grande.

1.4.2 Le Ou-Bruité

Un autre réseau très employé en diagnostic médical est le Ou-Bruité [ONI00]. Un 'Ou' est un réseau ayant la forme du réseau intuitif et tel que si l'un des nœuds symptômes vaut 1 alors le nœud diagnostic vaudra 1 aussi.
Pour le cas du 'Ou-bruité', il s'agit de mettre une étape de bruitage, suivi d'un 'Ou' classique comme sur la figure 4.

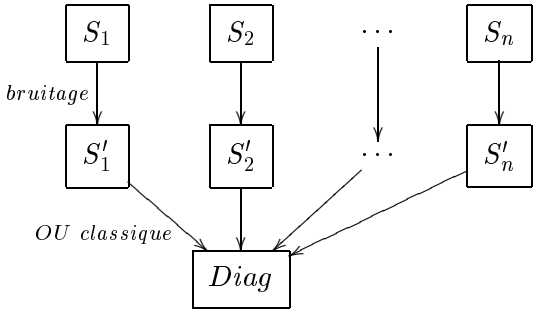


FIG. 4 – le 'Ou-bruité'.

Cela consiste à prendre une probabilité p telle que si le nœud vaut 1 alors le nœud bruité correspondant vaille 1 avec la probabilité p.

Comment faire lorsque les variables utilisées sont continues ?
Cet aspect peut être pris en compte de différentes façons :

- Soit en discrétisant les variables [MAT00],
- Soit en faisant une hypothèse sur la forme de la distribution (ici gaussienne). Ainsi les paramètres à évaluer sont les paramètres d'une distribution continue, au lieu d'être les probabilités individuelles de chaque valeur discrète.

Lorsqu'on utilise un modèle conditionnel gaussien, lors de l'inférence, on propage un potentiel lié à la moyenne et l'écart-type de la gaussienne au lieu de propager un potientiel lié à la table de probabilité.

On généralise alors les opérations de potentiels : extension, restriction, multipication, division et marginalisation au cas de nœuds continus (cf [LAU99]).

Un bémol existe cependant, il est impossible de faire de l'inférence *exacte* lorsque qu'un nœud discret D possède un parent gaussien G. En effet, la table de probabilité de D devrait alors contenir toutes les valeurs $\mathbb{P}(D = d/G = g)$ or G peut prendre une infinité de valeurs.

1.4.4 Les modèles *multi-conditionnels-gaussiens*

Dans de nombreux cas, une variable ne peut pas être modélisée par une gaussienne, on la remplace donc par une combinaison de plusieurs lois gaussiennes. Pour cela on utilise une structure particulière faisant intervenir un nœud discret caché H dont le nombre d'états symbolisera le nombre de gaussiennes que l'on veut utiliser (voir figure 5).

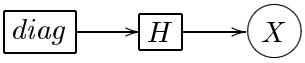


FIG. 5 – Comment combiner plusieurs gaussiennes.

1.5 La Bayes Net Toolbox

La Bayes Net Toolbox (BNT) est une boîte à outils gratuite programmée pour Matlab 5 principalement par K. MURPHY [MUR00]. Elle s'avère être un très bon outil pour l' utilisation des réseaux bayésiens car :

- elle effectue l'inférence bayésienne soit de manière exacte, soit de manière approchée, avec plusieurs moteurs implémentés,
- elle gére également les distributions de probabilités discrètes, gaussiennes et les 'Oubruités' binaires,
- elle traite l'apprentissage des paramètres,
- elle gére l'apprentissage de la struture avec IC*, PC et K2,

De plus cet outil récupère les avantages de Matlab, qui a de bons algorithmes numériques, et les tableaux sont particulièrement bien adaptés à la gestion des tables de probabilités conditionnelles des réseaux bayésiens.

2 Apprentissage des paramètres à partir de données

2.1 Avec des données complètes

2.1.1 Approche fréquentielle

Le but, à présent, va être de remplir les tables de probabilités conditionnelles. La première méthode consiste à utiliser des probabilités subjectives qui auront pu être données par un expert, par exemple. Cependant cette méthode est très risquée car, souvent, par

résultats ultérieurs, fournis par l'inférence dans le réseau bayésien, dépendent énormément de ces paramètres.

Une autre méthode, bien plus satisfaisante quand elle est réalisable, consiste à évaluer les lois jointes à partir d'une base de données. Il s'agit d'estimer les probabilités en utilisant :

$$\mathbb{P}(X = x|Y = y, Z = z) = \frac{\mathbb{P}(X = x, Y = y, Z = z)}{\mathbb{P}(Y = y, Z = z)}$$

après estimation de $\mathbb{P}$ par *maximum de vraisemblance*, on trouve

$$\mathbb{P}(X = x|Y = y, Z = z) \simeq \frac{N_{X=x,Y=y,Z=z}}{N_{Y=y,Z=z}} \tag{4}$$

où N_{cas} représente le nombre de fois que le *cas* est représenté dans la base de données.

⋇ Cette fonction est implémentée sous BNT, la fonction **learn_params** prend en entrée le réseau et une base de données pour effectuer l'apprentissage.

2.1.2 Approche bayésienne (avec *à priori*)

Soit X une variable discrète ayant r états possibles $x^1, x^2 \cdots, x^r$.
Soit $\theta_k = \mathbb{P}(X = x^k|\theta, \eta)$ les paramètres à évaluer, et $N_1, N_2, \cdots, N_r$, ou N_k est le nombre de fois que $X = x^k$ dans notre base de données. Soient $\alpha_1, \alpha_2 \cdots \alpha_r$ tous strictement positifs. Le paramètre θ_k a une *distribution de Dirichlet d'exposants* $\alpha_1, \alpha_2, \cdots, \alpha_r$ quand sa densité de probabilité est donnée par :

$$\mathbb{P}(\theta_k|\eta) = \frac{\Gamma(\sum_{i=1}^{r} N_i)}{\prod_{i=1}^{r} \Gamma(N_i)} \prod_{i=1}^{r} \theta_i^{\alpha_i - 1}$$

où Γ est la fonction Gamma, qui satisfait $\Gamma(x + 1) = x\Gamma(x)$ et $\Gamma(1) = 1$.

On s'aperçoit que la manière d'évaluer la distribution de probabilité $\mathbb{P}$ par *maximum à posteriori* lorsqu'une telle distribution est mise à priori sur les paramètres est donnée par la formule suivante,

$$\mathbb{P}(X = x|Y = y, Z = z) \simeq \frac{N_{X=x,Y=y,Z=z} + \alpha_{X=x,Y=y,Z=z}}{N_{Y=y,Z=z} + \alpha_{Y=y,Z=z}} \tag{5}$$

On remarque alors que les α_k jouent un rôle d'occurences '*à priori*' de $X = x^k$ avant que l'on ait connaissance de la base d'exemples.

2.2 Avec des données manquantes

Que l'on choisisse l'approche fréquentielle ou bayésienne, on évalue (presque) toujours les paramètres grâce à l'algorithme EM car il est simple et efficace.

Son principe réside en deux étapes (voir table 1).

Soit $X_{observees}$ l'ensemble des variables observées.

L'algorihme EM possède des propriétés de convergence vers un optimum local. Dans le cas où de nombreuses données sont manquantes, il peut exister plusieurs optima locaux et il y a alors peu de chance pour que l'optimum atteint soit global.

Remarquons que cet algorithme fournit, après convergence, une valeur des paramètres et non une distribution pour ces paramètres.

⋇ Cet algorithme a été implémenté sous BNT dans la fonction **learn_params_em**.

(on peut fixer des à priori de Dirichlet).

$$\theta_{i,j,k}^{(0)} = \mathbb{P}(X_i = x_k / pa(X_i) = x_j)$$

Répéter

Expectation : On utilise les paramètres courants pour estimer les données manquantes.

$$E[N_{i,j,k}] = \sum_{l=1}^{n} \mathbb{P}(X_i = x_k / pa(X_i) = x_j, \theta_{i,j,k}^{(t)}, X_{observees})$$

Maximisation : Puis on calcule les nouveaux paramètres par max de vraisemblance (ou max à posteriori) avec ces valeurs sur l'échantillon complet.

$$\theta_{i,j,k}^{(t+1)} = \frac{E[N_{i,j,k}] \ (+\alpha_{x_i,x_j,x_k})}{\sum_k E[N_{i,j,k}] \ (+\alpha_{x_i,x_j,x_k})}$$

Tant que l'on n'est pas assez proche de l'optimum.

TAB. 1 – L'algorithme EM de Dempster

3 Apprentissage de la structure

3.1 Rappel de quelques notions

3.1.1 Le test du χ^2

Pour tester si un lien de dépendance causale est bien en adéquation avec notre base de données on va être amené à faire un test du χ^2, je vais donc expliquer brièvement en quoi un tel test consiste.

Le test du χ^2 concerne uniquement les lois discrètes, mais on peut l'utiliser aussi pour des échantillons continus regroupés en classes. Le modèle de base est toujours un èchantillon $(X_1, ..., X_n)$ d'une loi inconnue. Les classes, notées $c_1, ..., c_r$ sont une partition de l'ensemble des valeurs possibles. L'hypothèse à tester porte sur les probabilités des classes, pour lesquelles on se donne des valeurs théoriques $P_0(c_1), ..., P_0(c_r)$.

$$\mathcal{H}_0 : \mathbb{P}[X_i \in c_k] = P_0(c_k), \ \forall k = 1, ..., r \, .$$

Sous l'hypothèse $\mathcal{H}_0$, la distribution empirique de l'échantillon sur les classes doit être proche de la distribution théorique. La distribution empirique est fréquentielle :

$$P_1(c_k) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{c_k}(X_i) \ .$$

On mesure alors l'adéquation de la distribution empirique à la distribution théorique par la distance du χ^2 définie par

$$\mathcal{D}_{\chi^2}(P_0, P_1) = \sum_{h=1}^{r} \frac{(P_1(c_h) - P_0(c_h))^2}{P_0(c_h)}. \tag{6}$$

Remarquons que cela n'est pas une distance au sens usuel, elle n'est pas symétrique.

Théorème 3.1 *Sous l'hypothèse $\mathcal{H}_0$, la loi de la variable aléatoire $n\mathcal{D}_{\chi^2}(P_0, P_1)$ converge quand n tend vers l'infini vers la loi du χ^2 de paramètre r-1.*
Si l'hypothèse est fausse, alors la variables aléatoire $n\mathcal{D}_{\chi^2}(P_0, P_1)$ tend vers l'infini.

moins de l'ordre de la centaine.

Exemple d'application : *le test du χ^2 de contingence*
C'est un cas particulier du test du χ^2 permettant de tester l'indépendance de deux caractères.
Voici un exemple de deux caractères binaires, concernant des malades, pour lesquels on a observé s'ils ont ou non une tendance suicidaire. Leurs maladies ont été classées soit en "psychose", soit en "névrose". On souhaite savoir s'il y a une dépendance entre les tendances suicidaires et le classement des malades. Supposons que la table de contigence observée soit :

	tendance suicidaire	sans tendance	total
Psychose	20 (1/20)	180 (9/20)	200 (1/2)
Névrose	60 (3/20)	140 (7/20)	200 (1/2)
Total	80 (2/10)	320 (8/10)	400

Supposons que les deux caractères soit indépendants, c'est-à-dire que les lois marginales ne diffèrent pas trop des lois empiriques, ou encore, ce qui est équivalent, que *les fréquences conjointes doivent être proches des produits de fréquences marginales*

$$\mathcal{H}_0 : \ \mathbb{P}(X,Y) = \mathbb{P}(X)\mathbb{P}(Y).$$

$$\mathcal{D}_{\chi^2} = \frac{(\frac{1}{20} - \frac{1}{2}\frac{2}{10})^2}{\frac{1}{2}\frac{2}{10}} + \frac{(\frac{9}{20} - \frac{1}{2}\frac{8}{10})^2}{\frac{1}{2}\frac{8}{10}} + \frac{(\frac{3}{20} - \frac{1}{2}\frac{2}{10})^2}{\frac{1}{2}\frac{2}{10}} + \frac{(\frac{7}{20} - \frac{1}{2}\frac{8}{10})^2}{\frac{1}{2}\frac{8}{10}} = 0,0625$$

donc $n\mathcal{D}_{\chi^2} = 25$ que l'on doit comparer à la loi $\chi^2(1)$ (à un degré de liberté, en effet $(2-1)(2-1) = 1$).

$1 - \mathcal{F}_{\chi^2(1)}(25) \simeq 10^{-7}$ où $\mathcal{F}_{\chi^2(1)}$ désigne la fonction de répartition de $\chi^2(1)$,

$$\mathcal{F}_{\chi^2(n)}(u) = \frac{1}{2\Gamma(\frac{n}{2})} \int_0^u e^{-\frac{t}{2}} \left(\frac{t}{2}\right)^{\frac{n}{2}-1} dt \tag{7}$$

or comme $25 > 10^{-7}$, (c'est ce test que l'on utilise pour évaluer si la variable $n\mathcal{D}_{\chi^2}$ tend vers l'infini) on rejette l'hypothèse $\mathcal{H}_0$, et on en déduit qu'il y a une dépendance entre la tendance suicidaire et la classification des maladies.

Remarque 1 ***a°)*** *Pour le cas où l'on ne possède pas de grande base de données (entre 10 et 80 enregistrements), on peut tout de même utiliser le test du χ^2 via* le terme correctif de Yates, *on remplace alors $\frac{(P_1(c_h)-P_0(c_h))^2}{P_0(c_h)}$ par $\frac{(|P_1(c_h)-P_0(c_h)|-0,5)^2}{P_0(c_h)}$.*

b°) *Certains préfèrent utiliser* le test du G^2 (par exemple [SPI00]), *qui suit également une loi du χ^2 mais dont l'expression est*

$$G^2(P_0, P_1) = \sum_{h=1}^{r} P_1(c_h) \ln \frac{P_1(c_h)}{P_0(c_h)}. \tag{8}$$

3.1.2 La log-vraisemblance

Définition 6 *Soit $X = X_1, ..., X_n$, un vecteur de n variables aléatoires, suivant une distribution de probabilité $\mathbb{P}$, soit $\boldsymbol{D}$ une base de données de différentes valeurs prises par ces variables simultanemment, $x = (x_1, ..., x_n)$ et $N(x)$ le nombre d'exemples dans la base de données où $X_1 = x_1, ..., X_n = x_n$. Alors la vraisemblance est définie par :*

$$V(\mathbb{P}, \boldsymbol{D}) = \prod_{x \in X} \mathbb{P}(x)^{N(x)} \tag{9}$$

appliquer cette définition en utilisant les *paramètres issus d'un réseau bayésien* B, de plus, pour une plus grande stabilité numérique, on utilise souvent la *log-vraisemblance* qui est alors donnée par :

$$L(B, \boldsymbol{D}) = -\sum_{x \in X} N(x) \sum_{i=1}^{n} \log\left(\mathbb{P}(x_i / x_{pa(X_i)})\right) \tag{10}$$

où $x_\emptyset = \emptyset$ *et* $x_{\{a\} \cup E}$ signifie $X_a = x_a$ *et* x_E.

3.1.3 Description de l'exemple

Nous allons faire tourner pas à pas différents algorithmes d'apprentissage de la structure. Pour cela nous allons nous servir du célèbre exemple de diagnostic de la *dyspnée*, qui est en fait un problème pulmonaire. (cf graphe de la figure 6).

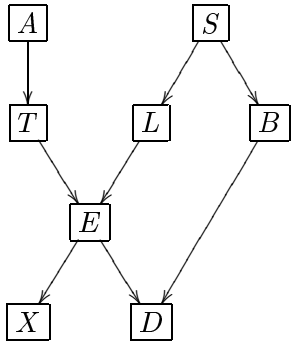


FIG. 6 – Le graphe de Lauritzen et Spiegelhalter.

Tous les nœuds sont ici des variables binaires et :

- A représente le fait que l'individu a été faire un voyage en Asie ou non ces derniers temps,
- S nous dit si le patient est fumeur ou non,
- T, s'il est atteint de la tuberculose,
- L, s'il a un cancer du poumon,
- B, s'il a une bronchite,
- E est un nœud qui fait le 'OU' logique entre T et L,
- X nous dit si la radiographie X des poumons du patient est saine ou s'il y a des traces suspectes,
- et D est le nœud diagnostic, à savoir si le patient est atteint de *Dyspnée* ou non ?

On peut remarquer que l'arc entre A et T nous dit que le fait d'avoir séjourné en Asie modifie le risque d'avoir contracté une tuberculose. On voit aussi qu'une tuberculose n'est pas toujours détectée aux rayons X.

3.2 Généralités

Il existe plusieurs approches pour faire de l'apprentissage de la structure :

- La première classe d'algorithmes évalue les dépendances causales à l'aide d'un test statistique (généralement χ^2 ou G^2), et ainsi ajouter ou supprimer des liens. Ensuite on recherche les orientations à mettre sur ces liens. Pour cela on cherche des V-structures, toujours à l'aide des dépendances causales. Cependant cette méthode ne permet pas de trouver un graphe orienté, mais seulement partiellement orienté.

une instantiation du résultat obtenu.

- La seconde classe d'algorithmes utilise un mécanisme différent, il consiste à parcourir l'ensemble des réseaux bayésiens et à évaluer si le réseau courant est bien adapté à nos données, en lui attribuant un score, puis après avoir défini un voisinage, voir si aucun autre réseau de ce voisinage ne possède un meilleur score, et ainsi de suite.
- La dernière classe d'algorithmes étudiés ici a un principe proche de ceux de la deuxième classe, sauf que cette fois on parcourt l'ensemble des équivalents de Markov.

Pour que ces deux dernières classes d'algorithmes soient efficaces, il faut que le score puisse être calculé rapidement. Or lorsque que l'on parcourt notre ensemble de référence, on passe d'un graphe à un de ces voisins donc une grande partie des liens ne change pas. Il faut donc pouvoir calculer le score localement.

Définition 7 *Un critère de score S est dit* décomposable *ou* local *s'il peut être ecrit comme une somme de mesures qui sont chacune fonction seulement du nœud et de ses parents.*

c'est-à-dire si n est le nombre de nœuds du graphe le score doit être de la forme suivante :

$$S(\mathcal{G}) = \sum_{i=1}^{n} s(X_i, pa(X_i))$$

3.3 Les algorithmes de causalité

3.3.1 L'algorithme IC/IC*

L'algorithme IC a été introduit par Pearl et Verma [PEA91] et est décrit plus précisément dans leur dernier ouvrage [PEA00]. Soit X l'ensemble des variables connues (et nœuds par abus de language).

La variante IC* de cet algorithme, permet de détecter si éventuellement il peut y avoir des variables latentes.

Ces algorithmes prennent en entrée une méthode pour évaluer les dépendances conditionnelles, le nombre de nœuds et éventuellement le nombre maximum de parents admissibles pour un nœud, et ont pour sortie un D.A.G .

⋇ Cet algorithme est accessible dans la Bayes Net Toolbox par l'intermédiaire de la fonction **learn_struct_pdag_ic_star**.

L'application

1°) Pour nous, cette étape sera simple car on va se servir du graphe de la figure 6 directement pour identifier les dépendances conditionnelles. L'algorithme utilise normalement un test statistique à partir d'une base de données pour déterminer s'il y a dépendance ou non. Cet exemple servant à comprendre les mécanismes mis en jeu par les algorithmes IC et IC*, il n'était alors pas nécéssaire de faire de tels tests.

En nous aidant des définitions et du théorème du paragraphe sur la d-séparation (page 5), on est capable de construire la table d'indépendance conditionnelle (partielle car vu le nombre de cas à traiter, j'ai pu en oublier, cependant j'espère qu'elle contient toute l'information) .

J'ai utilisé l'ordre d'énumération des variables suivants : A, S, T, L, B, E, X, D.

statistique, χ^2 ou G^2, pour évaluer s'il y a indépendance causale ou non).
Avec union, c'est-à-dire si $A \perp B | C$ et $A \perp B | (C, D)$ et $A \perp B | E$ alors $S_{AB} = S_{BA} = \{C, D, E\}$ même si $A \perp B | S_{AB}$ est faux.

2°) **Construction d'un graphe non orienté**
$\forall \{A, B\} \in X^2$, soit S_{AB} l'ensemble des variables telles que $A \perp B | S_{AB}$ est vrai. Si $S_{AB} = \emptyset$ alors on relie A et B, A—B.

3°) **Recherche des V-structures**
$\forall \{A, B\} \in X^2$, non adjacents, $\forall C \in X$, voisin de A et B, si $C \notin S_{AB}$ alors on crée une V-structure $A \rightarrow C \leftarrow B$.

4°) **Ajout récusif de flèches**
Répéter, sans créer de cycle ou de V-structure,
Si A—B et s'il existe un chemin orienté de A vers B alors on ajoute une flèche de A vers B.
Si A et B ne sont pas liés et s'il existe C tel que $A \rightarrow C$—B alors on oriente $C \rightarrow B$.
Si A—B et s'il existe deux chaînes A—$C \rightarrow B$ et A—$D \rightarrow B$ (ou $A \rightarrow D \rightarrow B$) avec C et D non adjacents alors on oriente $A \rightarrow B$.
Tant qu'il est possible d'ajouter des arcs.

5°) **Cause véritables (spécifique à l'algorithme IC*)**
Si A et B ne sont pas liés, $A \rightarrow C$ et $C \rightarrow B$ alors peut-être que le lien entre C et B est dans l'autre sens.

6°) **Instantiation de notre résultat**
Pour que notre algorithme rende un D.A.G. comme annoncé précédemment, il suffit d'orienter les arêtes non orientées de manière à ne pas ajouter de cycle ni de V-structure.

TAB. 2 – L'algorithme IC/IC* de Verma et Pearl.

$A \perp S$	oui
$A \perp S \mid (T, L, B)$	oui
$A \perp S \mid X$	non, X est un fils de E à arcs convergents
$A \perp S \mid (E, X, D)$	non
$A \perp T$	non
$A \perp L \mid (S, T)$	oui
$A \perp L \mid (E, X)$	non car ATEL est active
$A \perp L \mid D$	non car ATEDBSL est active
$A \perp L \mid B$	oui
$A \perp B \mid (S, T, L)$	oui
$A \perp B \mid E$	non car ATELSB est active
$A \perp B \mid D$	non car ATEDB est active
$A \perp B \mid (E, D, X)$	non
$A \perp (E, X, D)$	non
$A \perp (S, L, B, E, X, D) \mid T$	oui (T sépare A du reste) même avec les nœuds 1 par 1

$S \perp T \mid (A, L, B)$	oui
$S \perp T \mid (E, X, D)$	non car SLET est active
$S \perp L$	non et pour tout conditionnement
$S \perp B$	non et pour tout conditionnement
$S \perp E$	non
$S \perp E \mid L$	oui
$S \perp X$	non
$S \perp X \mid (T, L, B, E)$	oui
$S \perp X \mid D$	non car SBDEX est active

$S \perp D$	non
$S \perp D \mid T$	oui
$S \perp D \mid L$	oui
$S \perp D \mid B$	oui
$S \perp D \mid E$	oui
$S \perp D \mid X$	oui

$T \perp L \mid D$	non car TEDBSL est active
$T \perp B \mid (A,S,L)$	oui
$T \perp B \mid (E,X,D)$	non car TELSB est active
$T \perp E$	non et pour tout conditionnement
$T \perp X$	non
$T \perp X \mid E$	oui
$T \perp D$	non
$T \perp D \mid L$	non
$T \perp D \mid E$	non car TELSBD est active
$T \perp D \mid (L,E)$	oui

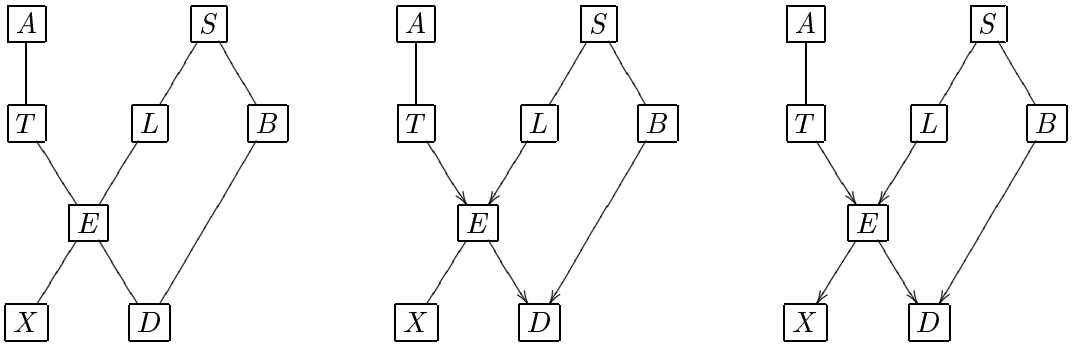
$L \perp B$	non
$L \perp B \mid (S,E,D)$	oui
$L \perp E$	non
$L \perp X \mid (E,D)$	oui
$L \perp B \mid S$	oui
$L \perp (E,X,D)$	non
$L \perp D \mid (E,S)$	oui

$B \perp (E,X,D)$	non
$B \perp E \mid S$	oui
$B \perp X \mid S$	oui
$B \perp X \mid L$	oui
$B \perp X \mid E$	oui
$E \perp (X,D)$	non
$X \perp D \mid E$	oui

2°) Les résultats de l'étape précédente nous permettent de créer les ensembles S_{AB} :

$S_{AS} = \{\text{T,L,B}\}$	$S_{ST} = \{\text{A,L,B}\}$	$S_{TB} = \{\text{A,S,L}\}$	$S_{LD} = \{\text{E,S}\}$
$S_{AT} = \emptyset$	$S_{SL} = \emptyset$	$S_{TE} = \emptyset$	$S_{BE} = \{\text{S}\}$
$S_{AL} = \{\text{S,T,B}\}$	$S_{SB} = \emptyset$	$S_{TX} = \{\text{E}\}$	$S_{BX} = \{\text{S,L,E}\}$
$S_{AB} = \{\text{S,T,L}\}$	$S_{SE} = \{\text{L}\}$	$S_{TD} = \{\text{L,E}\}$	$S_{BD} = \emptyset$
$S_{AE} = \{\text{T}\}$	$S_{SX} = \{\text{T,L,B,E}\}$	$S_{LB} = \{\text{S,E,D}\}$	$S_{EX} = \emptyset$
$S_{AX} = \{\text{T}\}$	$S_{SD} = \{\text{T,L,B,E,X}\}$	$S_{LE} = \emptyset$	$S_{ED} = \emptyset$
$S_{AD} = \{\text{T}\}$	$S_{TL} = \{\text{A,S,B}\}$	$S_{LX} = \{\text{E,D}\}$	$S_{XD} = \{\text{E}\}$

Donc, à l'issue de cette étape on trouve le graphe à l'étape 2, figure 7.



étape 2 étape 3 étape 4

FIG. 7 – Résultat de l'algorithme IC

Bien sûr, comme ici on s'est servi du graphe pour vérifier les cas de dépendances conditionnelles, on a trouvé tous les liens et aucun autre, or en raisonnant à partir des données, cela est rarement le cas.

3°) Recherchons les V-structures :
en suivant la règle de l'algorithme on trouve que
T et L ont un voisin commun, E, qui n'appartient pas à S_{TL} ={A,S,B} et
B et E ont un voisin commun, D, qui n'appartient pas à S_{BE}={S}
d'ou le graphe de l'étape 3, figure 7.

4°) La première règle ne nous permet pas d'ajouter des flèches. Par contre, pour la deuxième règle, on voit que L et X ne sont pas liés, et $L \rightarrow E$ et E–X donc on oriente $E \rightarrow X$ (cf figure 7 étape 4). La troisième règle est également stérile. Lors du deuxième passage, aucune flèche n'a pu être ajoutée donc on arrête là.

5°) Pour cette dernière étape, spécifique à l'algorithme IC*, on va introduire un nouveau type de lien, noté $A \leftrightarrow B$ quand on n'arrive pas à distinguer la cause de l'effet, c'est-à-dire $A \rightarrow B$ de $A \leftarrow B$ ou éventuellement du cas ou il existe une variable latente $A \rightarrow L \leftarrow B$. Et dans notre exemple cela va introduire un doute pour les liens 'EX' et 'ED', d'où le graphe de l'étape 5, figure 7. Cela veut en fait dire qu'il peut y avoir

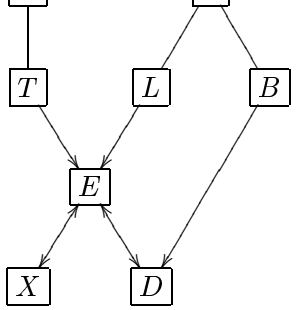


FIG. 8 – Résultat de l'algorithme IC*

une variable latente entre E et X ou entre E et D , mais s'il n'y en a pas, il faut se rappeller que XED ne peut pas former de V-structure.

6°) On peut remarquer, que bien que l'on soit parti avec une information maximale (le graphe lui-même), cet algorithme nous permet de retrouver l'équivalent de markov du graphe de départ. Mais le fait qu'il n'y ait pas de flèche sur un lien, nous indique certaines choses : d'une part les variables liées ne sont pas indépendantes, d'autre part, lorsque L—S—B (comme dans notre exemple), on peut dire que LSD ne forme pas une V-struture. De plus, on voit que l'étape 5, qui n'était pas dans l'algorithme initial, peut introduire du doute là où le résultat de l'étape 4 était bon, cette étape n'est donc à utiliser que lorsque l'on pense qu'il est possible qu'il existe des variables cachées. Pour finir, on oriente donc l'arête A–T de manière quelconque, et les arêtes L–S et S–B de manière à ne pas former de V-structure en S, car ici on ne peut pas introduire de cycle.

3.3.2 L'algorithme PC

L'algorithme PC a été introduit par Spirtes, Glymour et Scheines (1993) et revu lors de leur dernière publication ([SPI00]), son principe est très similaire à l'algorithme IC, il utilise un test statistique, χ^2 ou G^2, pour évaluer s'il y a indépendance causale ou pas, sauf qu'il part d'un graphe non orienté complet, et supprime les arcs.

L'algorithme PC prend en entrée une méthode pour évaluer les dépendances conditionnelles, le nombre de nœuds et éventuellement le nombre maximum de parents admissibles pour un nœud, et a pour sortie un D.A.G .

Comme pour l'algorithme précédent on peut faire une recherche de variables latentes en introduisant l'étape 5 de l'algorithme IC à cet algorithme qui devient alors l'*algorithme FCI*, non développé ici.

⋇ Cet algorithme est accessible dans la Bayes Net Toolbox par l'intermédiaire de la fonction **learn_struct_pdag_pc**.

L'application

1°) Comme pour les algorithmes IC et IC*, on n'effectue pas de test statistique ici, mais on utilisera le graphe figure 6 et le critère de d-séparation pour détecter les dépendances causales.

On commence avec le graphe complètement connecté.

n=0

Dans ce cas on vérifie les indépendances simples en nous référant au graphe d'origine (normalement il faut faire un test statistique à partir des données) ($A \perp S$ donc on

Soit C le graphe complet entre tous les nœuds de X.
n=0
Répéter
$\forall A, B \in X^2$ vérifiant A—B et le nombre de nœuds adjacents à A ($\neq B$) est supérieur à n.
Alors soit S un ensemble de nœuds adjacents à A ($\neq B$) de cardinalité n.
Si $A \perp B | S$ alors
– supprimer l'arc A–B,
– $SepSet(A, B) \leftarrow SepSet(A, B) \cup S$
– $SepSet(B, A) \leftarrow SepSet(B, A) \cup S$
n=n+1
jusqu'à $\forall (A, B) \in X^2$ l'ensemble des nœuds adjacents à A et différent de B soit de cardinalité strictement inférieure à n.

2°) **Recherche des V-structures**
$\forall (A, B, C) \in X^3$ tels que A et B ne sont pas liés et A—B—C,
si $C \notin Sepset(A, B)$ alors on crée une V-structure $A \rightarrow C \leftarrow B$.

3°) **Ajout récusif de flèches**
Répéter, sans créer de cycle ou de V-structure,
Si A—B et s'il existe un chemin orienté de A vers B alors on ajoute une flèche de A vers B.
Si A et B ne sont pas liés et s'il existe C tel que $A \rightarrow C$—B alors on oriente $C \rightarrow B$.
Si A—B et s'il existe deux chaînes A—$C \rightarrow B$ et A—$D \rightarrow B$ (ou $A \rightarrow D \rightarrow B$) avec C et D non adjacent alors on oriente $A \rightarrow B$.
Tant qu'il est possible d'ajouter des arcs.

4°) **Instantiation de notre résultat**
Pour que notre algorithme rende un D.A.G. comme annoncé précédemment, il suffit d'orienter les arêtes non orientées de manière à ne pas ajouter de cycle ni de V-structure.

TAB. 3 – L'algorithme PC de Spirtes, Glymour et Scheines.

supprime l'arc, A et T dépendants donc l'arc reste ...). De plus comme à chaque fois $|S| = 0$ les ensembles SepSet restent vides.

On conserva donc les liens suivants : AT, AE, AX, AD, SL, SE, SX, SD, SB, TE, TX, TD, LE, LX, LD, BD, EX, ED car il y a lien de cause à effet, et ceux-ci, LB, BE, BX, BD car $L \leftarrow S \rightarrow B$ et XD. D'ou le graphe figure 9.

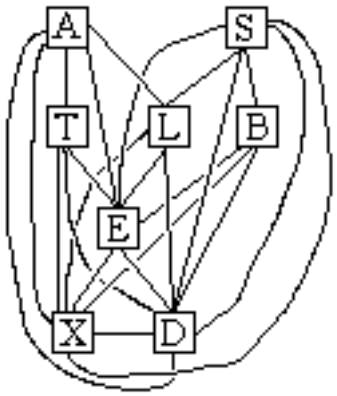


FIG. 9 – Algorithme PC, étape 1, n=0

plus quand on retire une arête, cela est effectif immédiatement, et donc les nœuds concernés ne seront plus voisins pour les tests ultérieurs.

n=1

A—T	:	$A \perp T \mid L$, $A \perp T \mid E$, $A \perp T \mid X$ et $A \perp T \mid D$ sont faux	
A—L	:	$A \perp L \mid T$ est vrai donc on retire l'arête A—L	SepSet(A,L)={T}
A—E	:	$A \perp E \mid T$ est vrai donc on retire l'arête A—E	SepSet(A,E)={T}
A—X	:	$A \perp X \mid T$ est vrai donc on retire l'arête A—X	SepSet(A,X)={T}
A—D	:	$A \perp D \mid T$ est vrai donc on retire l'arête A—D	SepSet(A,D)={T}
S—L	:	$S \perp L \mid B$, $S \perp L \mid E$, $S \perp L \mid X$ et $S \perp L \mid D$ sont faux	
S—B	:	tous les tests sont faux	
S—E	:	$S \perp E \mid L$ est vrai	SepSet(S,E)={L}
S—X	:	$S \perp X \mid L$ est vrai	SepSet(S,X)={L}
S—D	:	$S \perp D \mid L$ et $S \perp D \mid B$ sont faux	
T—A	:	$T \perp A \mid E$, $T \perp A \mid X$ et $T \perp A \mid D$ sont faux	
T—E	:	tous les tests sont faux	
T—X	:	$T \perp X \mid E$ est vrai donc l'arête disparait	SepSet(T,X)={E}
T—D	:	$T \perp D \mid A$ et $T \perp D \mid E$ sont faux	
L—S	:	$L \perp S \mid B$, $L \perp S \mid E$, $L \perp S \mid X$ et $L \perp S \mid D$ sont faux	
L—B	:	$L \perp B \mid S$ est vrai donc on retire l'arête L—B	SepSet(L,B)={S}
L—E	:	tous les tests sont faux	
L—X	:	$L \perp X \mid E$ est vrai	SepSet(L,X)={E}
L—D	:	$L \perp D \mid E$ et $L \perp D \mid S$ sont faux	
B—S	:	tous les tests sont faux	
B—E	:	$B \perp E \mid S$ est vrai	SepSet(B,E)={S}
B—X	:	$B \perp X \mid S$ est vrai	SepSet(B,X)={S}
B—D	:	$B \perp D \mid S$ est faux	
E—T	:	tous les tests sont faux	
E—L	:	tous les tests sont faux	
E—X	:	$E \perp X \mid T$, $E \perp X \mid L$ et $E \perp X \mid D$ sont faux	
E—D	:	aucun test réussi	
X—E	:	r.a.s.	
D—S	:	$D \perp S \mid T$,$D \perp S \mid L$ $D \perp S \mid B$ et $D \perp S \mid E$ sont faux	pas de retrait ici
D—T	:	$D \perp T \mid S$, $D \perp T \mid L$, $D \perp T \mid B$ et $D \perp T \mid E$ sont faux	ni la
D—L	:	$D \perp L \mid S$, $D \perp L \mid T$, $D \perp L \mid B$ et $D \perp L \mid E$ sont faux	pas de retrait
D—B	:	tous les tests sont faux	

On voit ici pourquoi cet algorithme est plus rapide que le précédent, car pour retirer un lien il suffit qu'un test d'indépendance soit faux, tandis que pour l'algorithme IC, pour ajouter un lien il faut que tous les tests soient vérifiés. D'où le graphe figure 10.

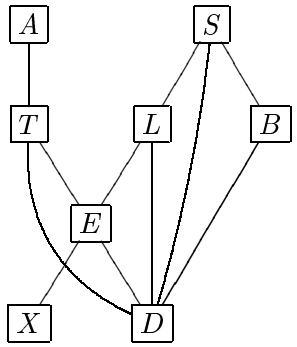


FIG. 10 – Algorithme PC, étape 1, n=1

On cherche à présent les nœuds ayant au moins 3 voisins :

n=2

S—L	:	$S \perp L \mid (B, D)$ est faux	
S—B	:	$S \perp B \mid (L, D)$ est faux	
S—D	:	$S \perp D \mid (L, B)$ est vrai	SepSet(S,D)={L,B}
L—S	:	$L \perp S \mid (B, D)$ est faux	
L—E	:	$L \perp E \mid (S, D)$ est faux	
L—D	:	$L \perp D \mid (S, E)$ est vrai	SepSet(L,D)={S,E}
E—?	:	ici tous les liens restent	
D—T	:	$D \perp T \mid (E, B)$ est vrai	SepSet(D,T)={E,B}

retrouvé le graphe de la figure 7, étape 2.

2°) Recherche des V-structures
T et L ne sont pas liés et T—E—L, or $E \notin SepSeT(T, L)$ donc on crée la V-structure $T \rightarrow E \leftarrow L$.
B et E ne sont pas liés et B—D—E, or $D \notin SepSeT(B, E)$ donc on crée la V-structure $B \rightarrow D \leftarrow E$.
Et on retrouve le graphe de figure 7, étape 3 de l'algorithme précédent.

3°) Comme cette étape est la même que l'étape 4 de l'algorithme PC, se referer à cette section (page 16).

4°) idem, voir page 17.

3.4 Les algorithmes de score dans l'espace des réseaux bayésiens

3.4.1 Généralités

On entre à présent dans une nouvelle catégorie d'algorithmes, au principe différent de ceux vus précédemment. Cette fois on ne cherche plus les dépendances causales, mais on parcourt l'ensemble des D.A.G, en évaluant chacun d'eux à l'aide d'une fonction de score.

Une première méthode pourrait utiliser simplement la *log-vraisemblance* comme score, seulement on ne satisferait alors pas le principe du rasoir d'Occam, c'est-à-dire qu'il ne faut pas rechercher l'efficacité au prix d'une trop lourde complexité, et trouver un bon compromis. Comme on le verra, le score bayésien ne respecte pas ce principe non plus.

Il a été montré que trouver le réseau de score maximum est un problème NP-difficile [CHI96] donc le parcours des graphes se fait souvent à l'aide d'une *métaheuristique* ou éventuellement d'un algorithme GLOUTON : on définit un voisinage de notre graphe courant, on prend celui qui à le meilleur score dans ce voisinage et on recommence jusqu'à ce que l'on soit dans un maximun local.

On peut d'abord remarquer que le nombre de structures différentes pour un réseau bayésien possédant n nœuds, $NS(n)$, est donné par la formule de récurrence suivante [ROB77], qui est super-exponentielle.

$$NS(n) = \begin{cases} 1 & , n = 0 \; et \; 1 \\ \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-1)} NS(n-i), & n > 1 \end{cases} \tag{11}$$

Ce qui donne $NS(2) = 3,\ NS(3) = 25,\ NS(5) = 29281,\ NS(10) \simeq 4,2 \times 10^{18}$.

On ne peut donc pas parcourir l'espace des réseaux bayésiens complètement, cela prendrait trop de temps. On utilise donc des heuristiques de recherche.

3.4.2 Le score *bayésien*, et l'algorithme K2

L'idée de l'algorithme K2 est de maximiser la probabilité de la structure sachant les données, pour cela on calcule cette probabilité pour plusieurs réseaux en utilisant la remarque suivante :

$$\frac{\mathbb{P}(B_{S_1}/D)}{\mathbb{P}(B_{S_2}/D)} = \frac{\frac{\mathbb{P}(B_{S_1}, D)}{P(D)}}{\frac{\mathbb{P}(B_{S_2}, D)}{P(D)}} = \frac{\mathbb{P}(B_{S_1}, D)}{\mathbb{P}(B_{S_1}, D)}$$

Pour calculer $\mathbb{P}(B_S, D)$, Cooper et Hersovits [COO92] ont donné le résultat suivant

Théorème 3.2 *Soit X l'ensemble de variables aléatoires* $\{X_1, ..., X_n\}$, $n \geqslant 1$, *où chaque* X_i *peut prendre les valeurs* $\{x_{i1}, ..., x_{ir_i}\}$, $r_i \geqslant 1$, $i = 1, ...n$. *Soit D la base de données et N le nombre de cas dans D, et soit* B_S *la structure du réseau sur X, et* $pa(X_i)$ *l'ensemble des*

D, $j = 1, ..., q_i$, $q_i \geqslant$, et N_{ijk}le nombre de cas dans D où X_i a la valeur x_{ik} et que $pa(X_i)$ est instantié en w_{ij}. Si $Nij = \sum k = 1^{r_i}$ alors

$$\mathbb{P}(B_S, D) = \mathbb{P}(B_S)\mathbb{P}(D|B) = \mathbb{P}(B_S) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{12}$$

où $\mathbb{P}(B_S)$ représente la probabilité à priori affectée au réseau B_S.

On remarque alors que si on prend un à priori uniforme, pour comparer le score de deux réseaux, on peut alors négliger ce terme.

L'équation 12 peut être vue comme une mesure de qualité du réseau par rapport aux données et est appelée la *mesure bayésienne*. ici elle est donnée lorsqu'il n'y a pas d'à priori sur les paramètres, mais il serait possible d'en mettre.

Comme on l'a vu précédemment, parcourir l'espace des réseaux bayésiens peut s'avérer super-exponentiel, pourtant Cooper et Hersovitz ont mis au point un algorithme de complexité $\mathcal{O}(n^3)$, grâce aux astuces suivantes.

On va d'abord supposer que la distribution à priori sur les structures est uniforme, puis on va imposer un ordre des nœuds, de manière qu'un nœud ne pourra être parent d'un autre que s'il précède celui-ci dans cet ordre. De plus on peut préciser un nombre maximum de parents pour chaque nœud (MaxP) si on le désire.

Cela va permettre de donner l'expresssion suivante :

$$\frac{1}{\mathbb{P}(B_S)} \max_S \mathbb{P}(B_S, D) = \prod_{i=1}^{n} \max_{pa(X_i)} \Big(\prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \Big)$$

On va donc pouvoir simplement calculer la qualité d'un jeu de parents pour un nœud par

$$score(X_i, pa(X_i)) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{13}$$

On peut à présent écrire l'algoritme (cf table 4).

Pour tout $i = 1..n$ **faire**
  $pa(X_i) = \emptyset$
  $Sc = score(X_i, pa(X_i))$
  Tant que *suite* et $|pa(X_i)| < MaxP$ **faire**
    `Chercher` Z précédant X_i qui maximise $score(X_i, pa(X_i) \cup \{Z\})$
    $nouvo = score(X_i, pa(X_i) \cup \{Z\})$
      Si $nouvo > Sc$ **alors**
        $Sc = nouvo$
        $pa(X_i) = pa(X_i) \cup \{Z\})$
      Sinon $suite = FAUX$
  Fin Tant que
Fin Pour

TAB. 4 – L'agorithme K2 (heuristique) de Cooper et Hersovits.

Etant donnée la forme de la fonction *score* cet algorithme ne marchera que pour les réseaux discrèts.

Malgré sa simplicité, l'algorithme K2 présente de très bonnes performances. De plus il peut être adapté aux distributions non uniformes, mais la mise en œuvre demande alors plus de calculs, et la plupart du temps, c'est cette version qui est utilisée.

⁎ Cet algorithme est implémenté dans BNT sous la fonction **learn_struct_K2**,

trés proche du score bayésien. En partant d'un à priori sur les structures, le but est d'estimer la probabilité à postériori $\mathbb{P}(B_S/D)$. *Ce score est meilleur que le score bayésien car sous certaines conditions il sera* 'score équivalent' (cf [CHI96]), *ce qui n'est pas le cas du score bayésien.*

L'application On va maintenant appliquer l'algorithme du K2 à notre exemple, en utilisant l'ordre d'énumération naturel du graphe : A, S, T, L, B, E, X, D. Pour calculer le score, j'ai du créer une base de données (de 3000 enregistrements) à partir du réseau bayésien d'origine [LAU88].

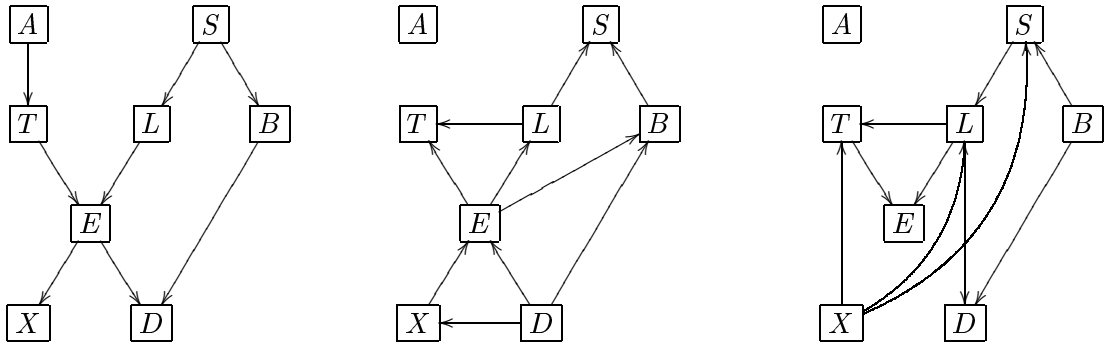
On démarre avec le graphe sans aucune arête, et on cherche quels parents peuvent être intéréssants pour les nœuds.

```
noeud A,  score(A,{}) = -190,3571
 il est le premier dans l'ordre d'énumération, il n'a donc aucun parent possible.
        pa(A)={}

noeud S,  score(B,{}) = -2083,4546
    test d'ajout de A à la liste de parents de S, score(S,{A}) = -2084,5793
 Si A est le parent de S le score baisse donc S n'aura pas de parent.
        pa(S)={}

noeud T,  score(T,{}) = -190,3571
    test d'ajout de A à la liste de parents de T, score(T,{A}) = -188,6059
    test d'ajout de S à la liste de parents de T, score(T,{S}) = -193,8942
 donc on ajoute définitivement A à la liste de parents de T, car cela monte le score à -188,6059.
    test d'ajout de S à la liste de parents de T, score(T,{A,S}) = -192,9130
        pa(T)={A}

noeud L,  score(L,{}) = -657,3157
    test d'ajout de A à la liste de parents de L, score(L,{A}) = -659,3218
    test d'ajout de S à la liste de parents de L, score(L,{S}) = -589,6113
    test d'ajout de T à la liste de parents de L, score(L,{T}) = -659,0556
 donc on ajoute de S à la liste de parents de L.
    test d'ajout de A à la liste de parents de L, score(L,{S,A}) = -593,1086
    test d'ajout de T à la liste de parents de L, score(L,{S,T}) = -592,9145
 on n'ajoute ni A ni T aux parents de L car le score n'est pas amélioré par rapport à -589,6113.
        pa(L)={S}

noeud B,  score(B,{}) = -2073,7370
    test d'ajout de A à la liste de parents de B, score(B,{A}) = -2075,7048
    test d'ajout de S à la liste de parents de B, score(B,{S}) = -1924,0400
    test d'ajout de T à la liste de parents de B, score(B,{T}) = -2074,9289
    test d'ajout de L à la liste de parents de B, score(B,{L}) = -2075,5092
 donc on ajoute de S à la liste de parents de B.
    test d'ajout de A à la liste de parents de B, score(B,{S,A}) = -1925,2598
    test d'ajout de T à la liste de parents de B, score(B,{S,T}) = -1926,4322
    test d'ajout de L à la liste de parents de B, score(B,{S,L}) = -1925,7966
        pa(B)={S}

noeud E,  score(E,{}) = -746,9125
    test d'ajout de A à la liste de parents de E, score(E,{A}) = -747,5882
    test d'ajout de S à la liste de parents de E, score(E,{S}) = -694,7437
    test d'ajout de T à la liste de parents de E, score(E,{T}) = -654,8509
    test d'ajout de L à la liste de parents de E, score(E,{L}) = -187,0493
    test d'ajout de B à la liste de parents de E, score(E,{B}) = -750,2057
 donc on ajoute de L à la liste de parents de E.
    test d'ajout de A à la liste de parents de E, score(E,{L,A}) = -186,1466
    test d'ajout de S à la liste de parents de E, score(E,{L,S}) = -192,4162
    test d'ajout de T à la liste de parents de E, score(E,{L,T}) = -10,6962
    test d'ajout de B à la liste de parents de E, score(E,{L,B}) = -191,9642
 donc on ajoute de T à la liste de parents de E.
    test d'ajout de A à la liste de parents de E, score(E,{L,T,A}) = -15,0420
    test d'ajout de S à la liste de parents de E, score(E,{L,T,S}) = -18,0390
    test d'ajout de B à la liste de parents de E, score(E,{L,T,B}) = -18,5681
        pa(E)={L,T}

noeud X,  score(X,{}) = -1041,6827
    test d'ajout de A à la liste de parents de X, score(X,{A}) = -1042,9976
```

```
    test d'ajout de T à la liste de parents de X, score(X,{T}) = -978,9178
    test d'ajout de L à la liste de parents de X, score(X,{L}) = -661,9965
    test d'ajout de B à la liste de parents de X, score(X,{B}) = -1045,1588
    test d'ajout de E à la liste de parents de X, score(X,{E}) = -578,6845
 donc on ajoute de E à la liste de parents de X.
    test d'ajout de A à la liste de parents de X, score(X,{E,A}) = -580,6359
    test d'ajout de S à la liste de parents de X, score(X,{E,S}) = -583,8023
    test d'ajout de T à la liste de parents de X, score(X,{E,T}) = -580,4251
    test d'ajout de L à la liste de parents de X, score(X,{E,L}) = -580,3877
    test d'ajout de B à la liste de parents de X, score(X,{E,B}) = -584,1088
        pa(X)={E}

noeud D,  score(D,{}) = -2070,0172
    test d'ajout de A à la liste de parents de D, score(D,{A}) = -2071,9848
    test d'ajout de S à la liste de parents de D, score(D,{S}) = -1987,0319
    test d'ajout de T à la liste de parents de D, score(D,{T}) = -2069,3745
    test d'ajout de L à la liste de parents de D, score(D,{L}) = -2031,8069
    test d'ajout de B à la liste de parents de D, score(D,{B}) = -1350,0418
    test d'ajout de E à la liste de parents de D, score(D,{E}) = -2030,5824
    test d'ajout de X à la liste de parents de D, score(D,{X}) = -2050,5831
 donc on ajoute de B à la liste de parents de D.
    test d'ajout de A à la liste de parents de D, score(D,{B,A}) = -1352,1835
    test d'ajout de S à la liste de parents de D, score(D,{B,S}) = -1351,1017
    test d'ajout de T à la liste de parents de D, score(D,{B,T}) = -1343,6423
    test d'ajout de L à la liste de parents de D, score(D,{B,L}) = -1278,5181
    test d'ajout de E à la liste de parents de D, score(D,{B,E}) = -1267,6625
    test d'ajout de X à la liste de parents de D, score(D,{B,X}) = -1300,5434
 donc on ajoute de E à la liste de parents de D.
    test d'ajout de A à la liste de parents de D, score(D,{B,E,A}) = -1269,7693
    test d'ajout de S à la liste de parents de D, score(D,{B,E,S}) = -1273,9731
    test d'ajout de T à la liste de parents de D, score(D,{B,E,T}) = -1269,5469
    test d'ajout de L à la liste de parents de D, score(D,{B,E,L}) = -1269,4956
    test d'ajout de X à la liste de parents de D, score(D,{B,E,X}) = -1273,2007
        pa(D)={B,E}
```

Et donc grâce aux listes de parents pour chaque nœud on arrive à reconstruire le graphe (a) de la figure 11.

Remarquons que si l'on avait pris un ordre différent, on aurait pû obtenir un graphe différent (cf figure 11 graphes b et c), de plus on voit que l'on arrive à obtenir un graphe qui n'est pas complétement connecté, ce qui indique le peu de pertinence de la variable qui dit si l'on est allé en Asie ou pas récemment.



(a) ordre=A,S,T,L,B,E,X,D (b) ordre=D,X,E,B,L,T,S,A (c) ordre=B,A,X,S,L,D,T,E

FIG. 11 – Différents graphes obtenus à l'aide de l'algorithme K2.

3.4.3 Les scores AIC et BIC

Toutes les fonctions de scores que l'on va voir à partir de maintenant, ont toutes la même forme : elles possèdent deux termes, le premier est la vraisemblance (simple à

par définir la *dimension* du modèle.

Soient X_i un nœud du réseau bayésien B de taille r_i et $pa(X_i)$ ses parents, alors le nombre de paramètres nécéssaires pour représenter la distribution de probabilité $\mathbb{P}(X_i/pa(X_i) = x_j)$ est égal à $r_i - 1$ donc pour représenter $\mathbb{P}(X_i/pa(X_i))$ il faudra

$$Dim(X_i, B) = (r_i - 1) \prod_{X_j \in pa(X_i)} r_j \tag{14}$$

Le nombre de paramètres nécessaires pour représenter toutes les tables de probabilités du réseau B est donc

$$Dim(B) = \sum_{X_i} Dim(X_i, B) \tag{15}$$

Remarquons que $dim(B)$ représente le nombre de probabilités indépendantes qu'il faut estimer pour obtenir les tables de probabilités associées au réseau B.

On peut à présent définir les fonctions de score AIC et BIC. Elles sont déduites de principes énoncés par Akaïke et Schwartz dans les années '70.

$$AIC(B, D) = \log \mathbb{P}(D|B, \theta^{MV}) - Dim(B) \tag{16}$$

$$BIC(B, D) = \log \mathbb{P}(D|B, \theta^{MV}) - \frac{1}{2} Dim(B) \log N \tag{17}$$

où θ^{MV} est la distribution des paramètres obtenue par 'max de vraissemblance' pour le réseau B.

Le terme $Dim(B)$ induit automatiquement le principe du rasoir d'Occam : *une structure avec peu d'arcs sera préférée à une structure ayant plus d'arcs tant que la vraisemblance du réseau le plus complexe sera plus basse ou égale à celle du réseau plus simple.*

On pourrait donc imaginer un algorithme, similaire à l'algorithme K2 qui utiliserait un de ces scores, car ils sont locaux.

⁎ La fonction **learn_struct_K2** peut utiliser le score BIC à la place du score bayésien.

3.4.4 Le score MDL et l'algorithme K3

En gardant les mêmes notations que pour le théorème 3.2, on peut définir le score basé sur le principe MDL (*Minimum Description Length*) :

$$MDL(B, D) = \log \mathbb{P}(B) + \log \mathbb{P}(D|B, \theta^{MV}) + \frac{1}{2}.Dim(B).\log N \tag{18}$$

où $\mathbb{P}(B)$ est la probabilité à priori qui est donnée au réseau bayésien B.

L'algorithme K2, quand il utilise le score MDL, qui est local, est appelé K3 [BOU94].

3.5 Les algorithmes de score sur les équivalents de Markov

3.5.1 Généralités

On a vu que l'ensemble des D.A.G était immense (équation 11) d'où l'idée de quotienter cet ensemble pour obtenir la notion d'espace des équivalents de Markov. Gillispie et Perlman [GIL01] ont mis en évidence que le rapport du nombre de D.A.G sur le nombre de classes d'équivalences de Makov à une asymptôte aux environs de 3,7.

Cet espace reste vaste, néanmoins il possède de meilleures propriétés que l'espace des réseaux bayésiens car là où une heuristique sur l'espace des RB peut boucler sur différentes structures équivalentes, une heuristique sur l'ensemble des équivalents de Markov a plus

équivalent, ce sera soit l'optimum, soit on continuera la recherche.

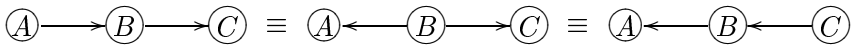
A présent, on est obligé d'utiliser des critères de score sur les équivalents. Or le critère bayésien utilisé dans l'algoritme K2 n'est pas *score équivalent* car l'a priori mis sur les structures n'est pas uniforme d'un point de vue des ensembles de relations causales que peut représenter un graphe, en effet le nombre de graphes représentant un tel ensemble n'est pas constant.

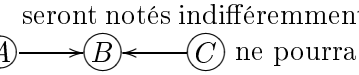
3.5.2 Comment trouver la classe d'équivalence d'un D.A.G

Définition 8 *Deux* D.A.G *sont dits* équivalents *(noté $\equiv$) s'ils induisent des lois jointes égales.*

Théorème 3.3 (Verma et Pearl, 1990) *Deux* D.A.G *sont équivalents si et seulement si ils ont le même squelette et les mêmes V-structures.*

Ainsi les trois réseaux équivalents suivants



seront notés indifféremment et sans ambiguité 

. Par contre le réseau 

ne pourra pas être noté différemment.

Définition 9 *L'ensemble des* équivalents de Markov *est défini par l'ensemble des graphes acycliques dirigés (*D.A.G*) quotienté par la relation d'équivalence $\equiv$.*

Définition 10 *Un critère de score qui associe une même valeur à deux graphes équivalents est dit '*score équivalent*'.*

Quand les structures des réseaux bayésiens sont interprétées comme des ensembles d'indépendances conditionnelles, tous les D.A.G d'une même classe d'équivalence de Markov représentent le même ensemble de contraintes. Il faut donc avoir un critère *score équivalent*pour faire de l'apprentisasge de structure dans l'espace des équivalents de Markov. Chickering [CHI96] a montré que les critères AIC, BIC, MDL et également le score BDe sous certaines conditions sont tous '*score équivalents*'.

Définition 11 *On dira qu'un arc est '*fixe dans le graphe $\mathcal{G}$*' s'il est orienté dans la même direction pour tous les graphes équivalents à $\mathcal{G}$.*
*Sinon on dira que c'est un arc '*réversible*'.*

Donc un arc sera fixe s'il ne fait pas partie d'une V-structure, ou si son renversement ne crée pas de V-structure.
Le P.D.A.G qui représentera la classe d'équivalence de Markov sera celui où tous ces arcs seront fixes, *et aucun autre*, ou autrement dit, qui n'aura aucun arc réversible.

3.5.3 Opérations sur les P.D.A.G

Pour pouvoir parcourir l'ensemble des équivalents de Markov, il faut avoir des opérations sur les P.D.A.G qui permettent de passer d'un représentant de Markov à un autre. De plus, il faut pouvoir mettre à jour le score aprés une opération élémentaire sans avoir à le recalculer entièrement, si l'on veut que notre algorithme soit efficace.

[CHI02] propose l'ensemble d'opérateurs décrits dans la table 5. Cet ensemble est complet, c'est-à-dire que tout P.D.A.G peut être atteint grâce à ces opérateurs. On peut remarquer que les opérateurs **ReverseD** et **MakeV** sont redondants.

Ces opérations peuvent nous faire sortir de l'ensemble des représentants des classes de Markov tel que nous les avons définis précédemment, donc il faut également disposer d'une fonction qui permette de passer d'un P.D.A.G quelconque à un représentant. Une fois que cela sera implémenté, nous pourrons disposer d'une méthode d'apprentissage de la structure dans l'espace des équivalents de Markov.

introduire un arc non-dirigé entre X et Y noté X—Y.
- **DeleteU :** On peut supprimer tout arc non-dirigé X—Y
- **InsertD :** Pour toute paire de nœuds X et Y qui ne sont pas adjacents dans le graphe, on peut introduire un arc dirigé de X vers Y.
- **DeleteD :** On peut supprimer tout arc dirigé entre X et Y.
- **ReverseD :** On peut renverser l'arc dirigé entre X et Y.
- **MakeV :** Pour tout triplet de nœuds X, Y et Z qui ne sont pas adjacents, ou contiennent les arc non-dirigés X—Y ou Y—Z, On peut créer une V-struture $X \rightarrow Y \leftarrow Z$.

TAB. 5 – Opérations élémentaires sur les P.D.A.G.

4 Cadre Applicatif : détection de cas de cancer de la thyroïde

J'ai pû appliquer les réseaux bayésiens sur un exemple d'aide au diagnostic médical, avec comme support le travail de Pierre MAHE durant son stage à l'INSA ainsi qu'une base de données d'apprentissage contenant 2800 enregistrements et d'une base de données de test possédant 972 enregistrements.

La taille modeste de la base de données de test va provoquer une instabilité lorsque le taux de rejet (voir page 28) sera élevé, en effet si ce taux à une valeur de 95%, le fait qu'**une** seule valeur soit mal évaluée va faire chuter le taux de bonne classification de plus de 2%.

4.1 Réseau naïf avec discrétisation des données continues

4.1.1 Formatage de la base de données

Pour l'implémentation, on commence par charger toutes les fonctions de la Bayes Net Toolbox grâce à la commande **add_BNT_to_path**. puis on sélectionne un à priori uniforme. Ensuite il faut reformater les bases de données dans deux tableaux matlab : apprent et test, pour lesquels on a retiré les colonnes 27, 28, 29, 30, 17, 19, 21, 23, 25 qui ne sont pas pertinentes pour notre réseau et on laisse de coté la colonne 30 qui code le diagnostic, d'où les 22 différentes variables de la table 6.

4.1.2 Echantillonnage des données continues

Pour échantillonner les données continues, on va se servir uniquement des lignes complètes des données d'apprentissage.

Pour avoir des variables optimales dans le sens d'un certain critère, il faut prendre en compte la perte d'information que provoque la discrétisation et la complexité du modèle que l'on construit. Il est bien évident que si l'on ne met qu'une seule valeur par classe on ne perd pas d'information, mais cette discrétisation est alors beaucoup trop complexe. Les différents critères utilisés vont donc avoir un terme de vraissemblance et un terme de pénalité.

Soit $A_1 \cdots A_q$ une partition d'une distribution inconnue λ, le but étant d'approximer λ avec un histogramme construit sur une sous-partition $C = B_1 \cdots B_c$ $(c \leq q)$. La distribution de probabilité λ_C ainsi construite à partir de C est une estimation de λ.

Ces différents critères sont déduits de principes proposés dans les années '70 par AKAIKE [AKA73]. Le score que l'on va affecter à la distribution λ_C ne dépend que de C et est de la forme suivante :

$$IC(C) = g(|C|) - \sum_{B \in C} \lambda_C \ln\left(\frac{\lambda_C(B)}{\nu_C(B)}\right)$$

1	âge (continue)
2	sexe (0=feminin et 1=masculin)
3	sous thyroxine
4	demande de la thyroxine
5	sous traitement antithyroide
6	a contracté une autre maladie
7	femme enceinte
8	chirurgie de la thyroide
9	sous traitement I131
10	demande d'hypothyroide
11	demande d'hyperthyroide
12	sous lithium
13	présence d'un goitre
14	présence d'une tumeur
15	hypopituitary
16	psych
17	valeur de TSH (continue)
18	valeur de T3 (continue)
19	valeur de TT4 (continue)
20	valeur de T4U (continue)
21	valeur de FTI (continue)

TAB. 6 – Les 22 variables de *Thyroid* utilisées

où g est la fonction de pénalité qui diffère d'un critère à l'autre, et où λ est supposée être une distribution de probabilité continue sachant la distribution de probabilité *à priori* ν. C'est-à-dire que la densité de probabilité de λ est donnée par $\mathcal{F}_\lambda = \frac{d\lambda}{d\nu}$. La distribution ν_C est alors l'estimation de ν dans le sens précédent.

Soit N la taille de l'échantillon.

1^{er} critère	AIC	$g(c) = \frac{2c-1}{N}$
2^{ieme} critère	AIC*	$g(c) = \frac{c(1+\ln N)}{N}$
3^{ieme} critère	ϕ *	$g(c) = \frac{c(1+\ln(\ln N))}{N}$

Pour discrétiser, on utilise la fonction **hist_ic** issue des travaux de F. El-Matouat [MAT00]. Elle rend alors la taille des nœuds discrets (nombre de classes à l'issue de la discrétisation) ainsi créés dans le tableau **nbbornes**. Cette fonction peut également prendre un quatrième critère, qui effectue une discrétisation naïve. On obtient alors deux *tableaux de cellules* avec lesquels on va pouvoir faire de l'apprentissage des paramètres grâce à l'algorithme **EM** (expectation-maximisation), disponible dans la toolbox BNT avec la fonction **learn_param_em** (cf table 1 page 11).

4.1.3 Création du réseau

On crée le réseau bayésien naïf grâce à la commande **mk_bnet**, qui prend en entrée une matrice d'adjacence, la taille des nœuds et les indices des nœuds discrets. Nous allons à présent pouvoir faire de l'apprentissage et de l'inférence dans le réseau de la figure 12.

4.1.4 Apprentissage des paramètres

Pour plus de détails concernant cette section se reporter à la section 2. Le code suivant nous permet de faire l'apprentissage des paramètres.

```
for j=1:N bnet.CPD{j}=tabular_CPD(bnet,j,'prior_type','dirichlet','dirichlet_type','unif') end;
engine = jtree_inf_engine(bnet); max_iter=15;
[bnet2, LL,engine2] = learn_params_em(engine,apprent_ok', max_iter);
```

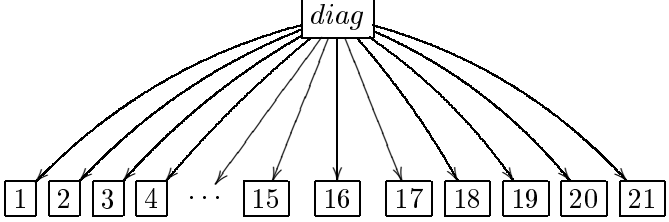


FIG. 12 – Le réseau bayésien naïf discrèt

Si l'on n'avait pas eu des paramètres manquants, on aurait pu utiliser la fonction **learn_params** à la place de **learn_params_em**.

4.1.5 Inférence bayésienne

Pour effectuer de l'inférence il suffit d'utiliser la fonction **enter_evidence** qui prend en entrée le moteur qui contient toutes les informations sur le réseau bayésien et un tableau de cellules qui contient les informations concernant les observations de certains nœuds. Pour de plus amples explications, se reporter à la section 1.3.

```
proba_post=zeros(Napp,2) ;
for i=1:Napp
   evidence=cell(1,N); evidence(2:N)=xechan_total2(i,:);
   [engine3, loglik]=enter_evidence(engine2,evidence);
   marg=marginal_nodes(engine3,diag); proba_post(i,:)=marg.T';
end
```

4.1.6 Matrices de confusion

Définition 12 *On dit que l'on prend une* décision de Bayes *(ou suivant le critère de Bayes) lorsque l'on choisit la solution ayant la plus forte probabilité, donc pour les variables bivalentes, la solution ayant une probabilité strictement supérieure à* $\frac{1}{2}$.
On dit que l'on prend une décision suivant une valeur de rejet r lorsque l'on choisit la solution ayant la plus forte probabilité seulement si celle-ci est supérieure à r.
Si aucune solution n'a de probabilité supérieure à r, on dit alors que ce cas est rejeté.

On peut donc afficher les résultats trouvés grâce à une matrice de confusion.

nombre de cas positifs bien classés	nombre de cas positifs mal classés	nombre de cas positifs rejetés
nombre de cas négatifs mal classés	nombre de cas négatifs bien classés	nombre de cas négatifs rejetés

```
[MMM COL]=max(proba_post,[],2); THEO=classe_apprent;
% on fait varier le taux de rejet et on récupère plusieurs matrices de
% confusion que l'on réunit dans un cell array pour ensuite les sauver.
valeurs_rejet=[0.5 0.55 0.6 0.65 0.7 0.75 0.8 0.85 0.875 0.9 0.92 0.93 0.94 0.95
0.955 0.96 0.965 0.97 0.975 0.98 0.9835 0.987 0.99 0.992 0.994 0.995 0.996
0.997 0.9975 0.998 0.9985 0.999 0.9992 0.9994 0.9996 0.9997 0.9998 0.9999 1]
Resultat=cell(3,3,length(valeurs_rejet));
for j=1:size(valeurs_rejet,1)
    Rejet=valeurs_rejet(j); M=zeros(3); COL(find(MMM<=Rejet))=3;       % REJET si Max(P) <= Rejet
       for i=1:size(THEO,1) M(THEO(i),COL(i))=M(THEO(i),COL(i))+1; end;
Resultat(:,:,j)=num2cell(M);
end
```

Ce code nous permet de recupérer les matrices de confusion de la table 7 (je rappelle que j'utilise ici le troisième critère de discrétisation).

4.1.7 Affichage de la courbe ROC

Les matrices de confusion précédentes nous permettent de fabriquer une courbe, appelée la courbe ROC. En abscisse on met le pourcentage de rejet, qui est ici défini par

17	43	0

décision de Bayes (r=0,5)

5	20	35

r=0,9

1	6	53

r=0,98

TAB. 7 – Quelques matrices de confusion.

$\frac{100 \times nbre\ de\ cas\ rejetes}{taille\ de\ la\ base\ de\ donnees}$. Et en ordonnés on met le pourcentage de points bien classés non rejetés, défini par $\frac{100 \times nbre\ de\ bien\ classes}{nbre\ de\ non\ rejetes}$.

Un médecin peut se servir de la courbe ROC en choisissant le pourcentage de bien classés qu'il veut atteindre, généralement 100%, alors ensuite il peut lire la valeur de rejet qui correspond à ce taux (par exemple pour la courbe bleue de la figure 13, cette valeur est de 86%, donc il sait que lorsque le classifieur donne un diagnostic, celui-ci est 'sûr', cependant dans 86% des cas celui-ci ne se prononcera pas et le médecin devra se fier uniquement à son avis personnel (ou éventuellement utiliser un autre classifieur).

Si le médecin choisit un pourcentage de bien classés de $99,4\%$, la courbe bleue atteint cette valeur pour un taux de rejet de 30% donc ici le médecin aura un classement presque sûr dans 70% des cas, il devra néanmoins étudier le cas personnellement pour retirer le doute d'un mauvais diagnostic.

On peut à présent afficher la courbe **ROC**, voir les courbes de la figure 13.

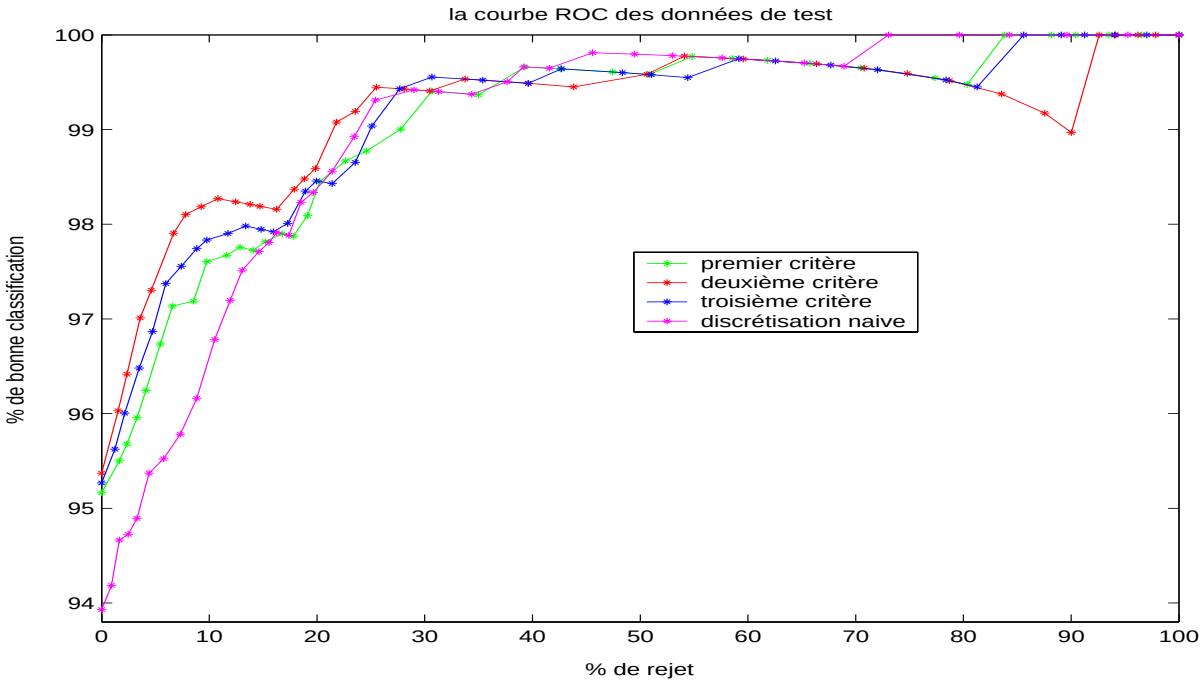


FIG. 13 – Résultats issus du réseau naïf discret pour les 3 critères de discrétisation.

On voit que quel que soit le critère de discrétisation que l'on utilise, le résultat est meilleur que pour la discrétisation naïve tant que le taux de rejet est inférieur à 20%. On va donc garder la courbe bleue, issue du troisième critère (qui est le critère par défaut) car même si elle est légèrement moins bonne que celle issue du deuxième critère, elle est plus stable lorsque le taux de rejet est élevé.

4.2.1 Apprentissage des paramètres

L'étape de création du réseau est la même que dans le cas précédent mais bien sûr on n'effectue pas d'échantillonnage et la taille des nœuds continus est 1.

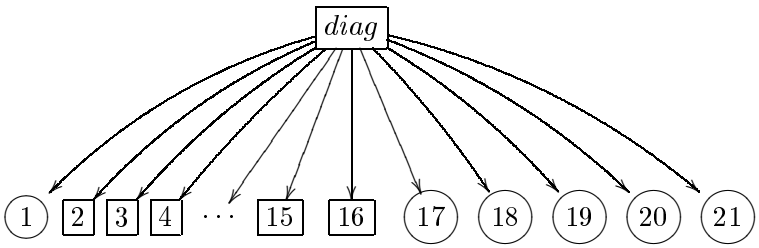


FIG. 14 – Le réseau bayésien naïf mixte

La différence par rapport au paragraphe 4.1.4 est qu'il faut utiliser la fonction **gaussian_CPD** pour les tables de probabilités des nœuds continus, les seuls changements sont donc :

```
for j=1:(length(discrets))
bnet.CPD{discrets(j)}=tabular_CPD(bnet,discrets(j),'prior_type','dirichlet','dirichlet_type',apriori);
end;
for k=1:length(continus)
bnet.CPD{continus(k)}=gaussian_CPD(bnet,continus(k),'cov_prior_entropic',1);
end;
```

4.2.2 Inférence et affichage de la courbe ROC

Le code ne subit ici que très peu de changements et on obtient les résultats de la figure 16. On remarque alors que la courbe du réseau mixte (noire) est légèrement meilleure que celle issue du troisième critère de discrétisation mais le temps de calcul est deux fois supérieur. Or cette courbe est moins stable lorsque le taux de rejet est élevé, cependant, on peut dire que les deux méthodes se valent.

4.3 Réseau avec apprentissage de la structure avec l'algorithme K2

4.3.1 Apprentissage

L'algorithme K2 ne marche qu'avec des données discrètes, donc on effectue ce travail comme dans la section 4.1.2.

Comme l'algorithme K2 ne marche que si aucune données n'est absente, on a écarté les entrées de la base de données qui possèdaient des valeurs manquantes. Le problème d'une telle méthode, est qu'elle modifie le nombre d'occurences relatives entre les différents cas qui peuvent se présenter.

Il est possible d'ajouter un état caché binaire avant chaque variable qui possède des données manquantes. Cet état caché modélise alors le fait qu'une valeur est présente ou non. L'inconvénient d'une telle modélisation est que, pour notre exemple, elle doublerait le nombre de nœuds.

La génération du réseau se fait grâce à la ligne de code suivante.

```
dag = learn_struct_K2(donnees_completes',node_sizes,[1:22],'max_fan_in',15);
```

Et on obtient le réseau bayésien de la figure 15. Pour plus de détails concernant le fonctionnement de l'algorithme K2, se reporter à la section 3.4.2

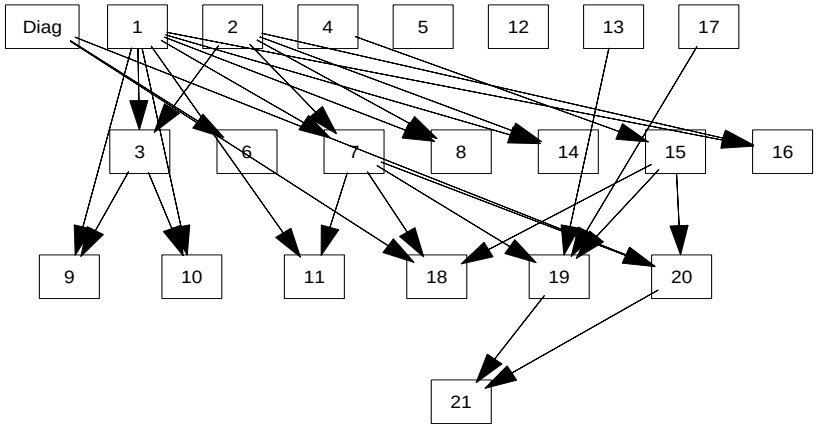


FIG. 15 – Réseau issu de K2 et de l'ordre d'énumération (diag,1,2,...,21)

4.3.2 Résultats

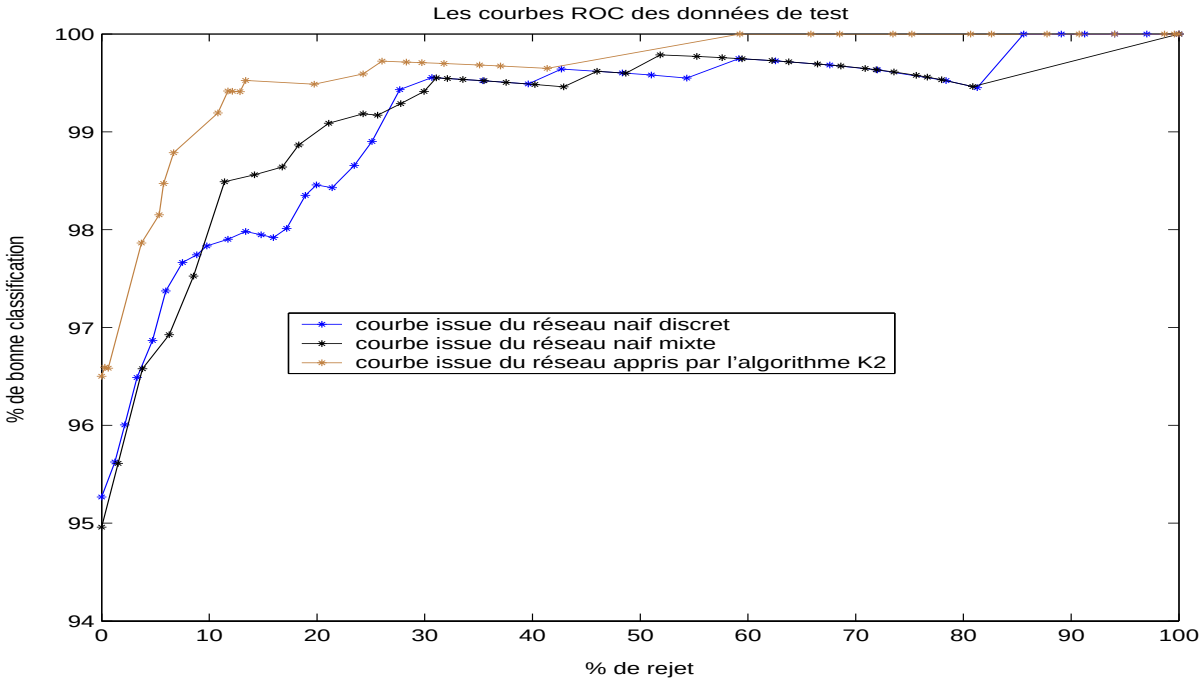


FIG. 16 – Résultats comparatifs entre les différentes techniques.

On voit que les courbes de la figure 16 sont trés bonnes lorsque le taux de rejet est autour de 30%, et il est intéressant de remarquer que la courbe issue de l'apprentissage de la structure atteint plus de $99,6\%$ de bonne classification dès 13% de rejet.

On peut remarquer que les résultats issus de réseaux ayant subis un apprentissage de la structure sont meilleurs que ceux avec des réseaux naïfs (pour les critères correspondants) sur les données de test, par contre ils sont plus mauvais sur les données d'apprentissage (voir figure 20). Comme on se sert des données d'apprentissage pour faire l'apprentissage des paramètres et de la structure, on obtient souvent de bons résultats sur ces valeurs, ce qui n'est pas significatif, car on peut apprendre '*par cœur*' les résultats pour ces données

Cependant je pense que ces résultats sont encore améliorables car l'ordre d'énumération qui a été introduit pour faire l'apprentissage a été choisi de manière simple (diag,1,...,21). Cet ordre influe très fortement sur le réseau qui va être créé par l'algorithme K2, et peut-être qu'avec un autre ordre on peut trouver un réseau qui soit plus adapté à notre problème. Néanmoins, j'en ai essayé quelques autres et pour l'instant je n'en ai pas trouvé de meilleur.

4.4 Réseau OU-bruité

J'ai essayé d'exploiter cette structure, seulement comme BNT ne gère que les nœuds binaires pour le OU-bruité, les résultats étaient médiocres. De plus, pour ré-implémenter les OU-bruités pour qu'il soit possible d'exploiter les nœuds d'arité quelconques s'avère être un travail fastidieux, et ce n'était pas l'objet de mon stage.

4.5 Modèle multi-conditionnel-gaussien

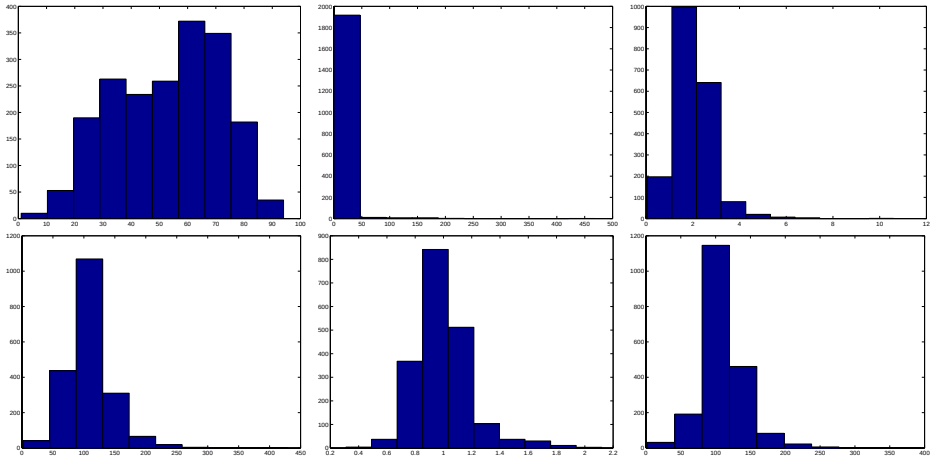


FIG. 17 – Histogramme des données continues.

Comme on peut le voir en traçant les histogrammes des variables continues (cf figure 17), celles-ci ne se modélisent pas bien avec des gaussiennes, donc comme BNT ne gère pour l'instant que les lois gaussiennes pour les variables continues, on peut ruser et utiliser des mélanges de gaussiennes en faisant intervenir un nœud discret juste avant le nœud gaussien, pour obtenir le réseau de la figure 18.

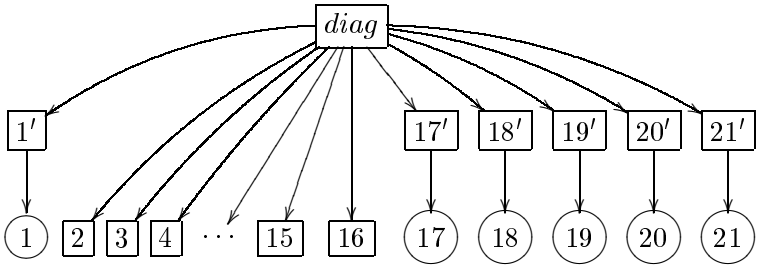


FIG. 18 – Le réseau bayésien naïf mixte

2 puis 3 gaussiennes pour modéliser les variables continues), j'ai obtenu les courbes de la figure 19.

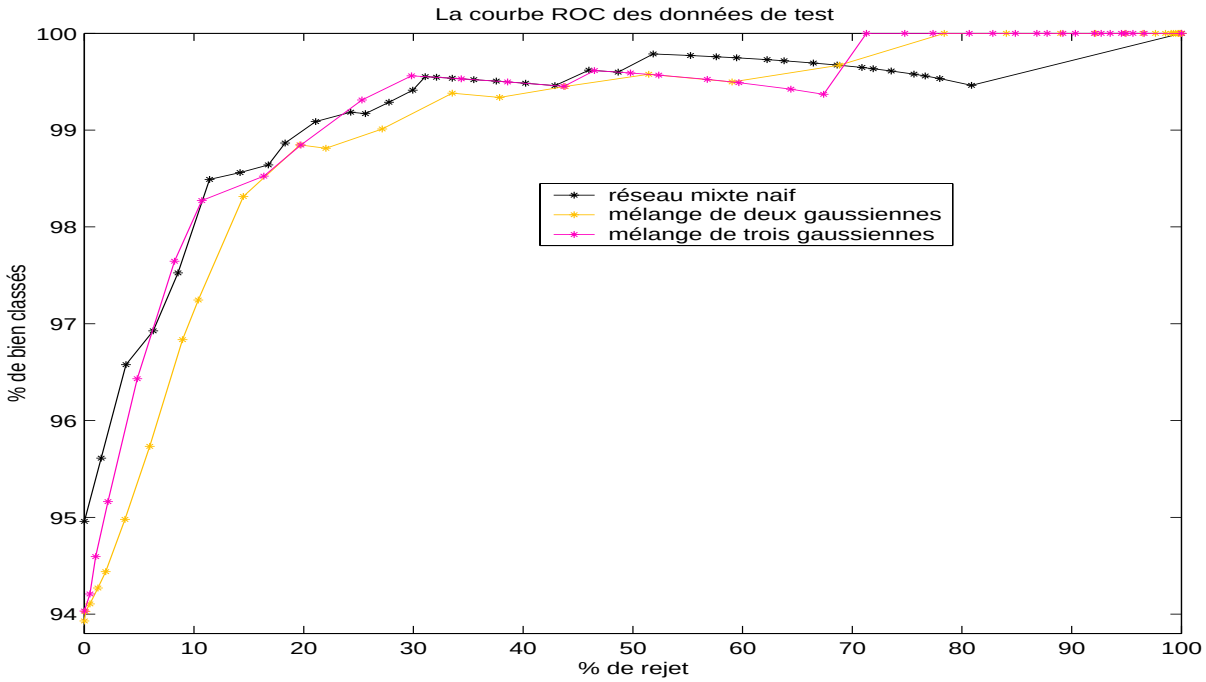


FIG. 19 – Résultats issus du réseau bayésien multi-gaussien.

On constate de légères améliorations, surtout au niveau de la fin de la courbe où on atteint un taux de bonne classification de 100% pour des taux de rejet de 65%, et 55%. Cependant l'introduction de nœuds supplémentaires par rapport au réseau mixte naïf, à doublé le temps de calcul.

4.6 Les durées de calcul des différentes méthodes

J'ai fait tourner les différents algorithmes sur un AMD ThunderBird 996Mhz avec 128Mo de mémoire SD-RAM 133Mhz (voir table 8).

Réseau naïf discrétisé avec le 1er critère	860 secondes
Réseau naïf discrétisé avec le 2ième critère	842 secondes
Réseau naïf discrétisé avec le 3ièm critère	858 secondes
Réseau naïf avec discrétisation naïve	1029 secondes
Réseau naïf mixte	1464 secondes
Réseau avec mélange de deux gaussiennes	4602 secondes
Réseau avec mélange de trois gaussiennes	5376 secondes
Réseau appris par l'algorithme K2	779 secondes

TAB. 8 – Comparatif des algorithmes en temps de calcul.

On remarque qu'au niveau de la qualité par rapport au temps de calcul, l'utilisation de l'algorithme K2 donne des résultats excellents, autant au niveau de l'estimation des paramètres que de l'inférence, ces deux étapes représentent d'ailleurs la plus grande partie du temps de calcul. On remarque également que lorsque l'on utilise une discrétisation

effet, l'algorithme EM pour l'apprentissage des paramètres effectue plus de boucles.

J'ai également essayé de faire tourner les algorithmes IC* et PC en utilisant le test du χ^2 pour évaluer les dépendances causales, mais ces méthodes se sont avérées beaucoup trop exigentes en temps de calcul et/ou en capacité mémoire pour ma machine, et cela même en réduisant significativement la base de données de tests.

4.7 Les limites de ces modèles

Un piège dans lequel il ne faut pas tomber est le problème du sur-apprentissage. Il est possible de trouver un modèle qui colle parfaitement aux données d'apprentissage, et si l'on se sert de ces mêmes données pour tracer les courbes ROC, on a généralement de très bons résultats, or ceux-ci ne sont pas significatifs. d'où l'intérêt d'utiliser des données de tests pour faire l'inférence, et tracer les courbes. J'en ai donc profité pour faire augmenter la sensibilité de l'apprentissage de paramètres. En effet l'algorithme EM tourne tant que $\frac{\Delta(vraisemblance)}{vraisemblance\ moyenne} < \epsilon$, or par défaut $\epsilon = 10^{-3}$, je l'ai donc abaissé à 10^{-5} pour tracer les courbes suivantes, et voir s'il y avait un sur-apprentissage. Ce changement a provoqué deux fois plus d'étapes EM, et a donné les courbes de la figure 20.

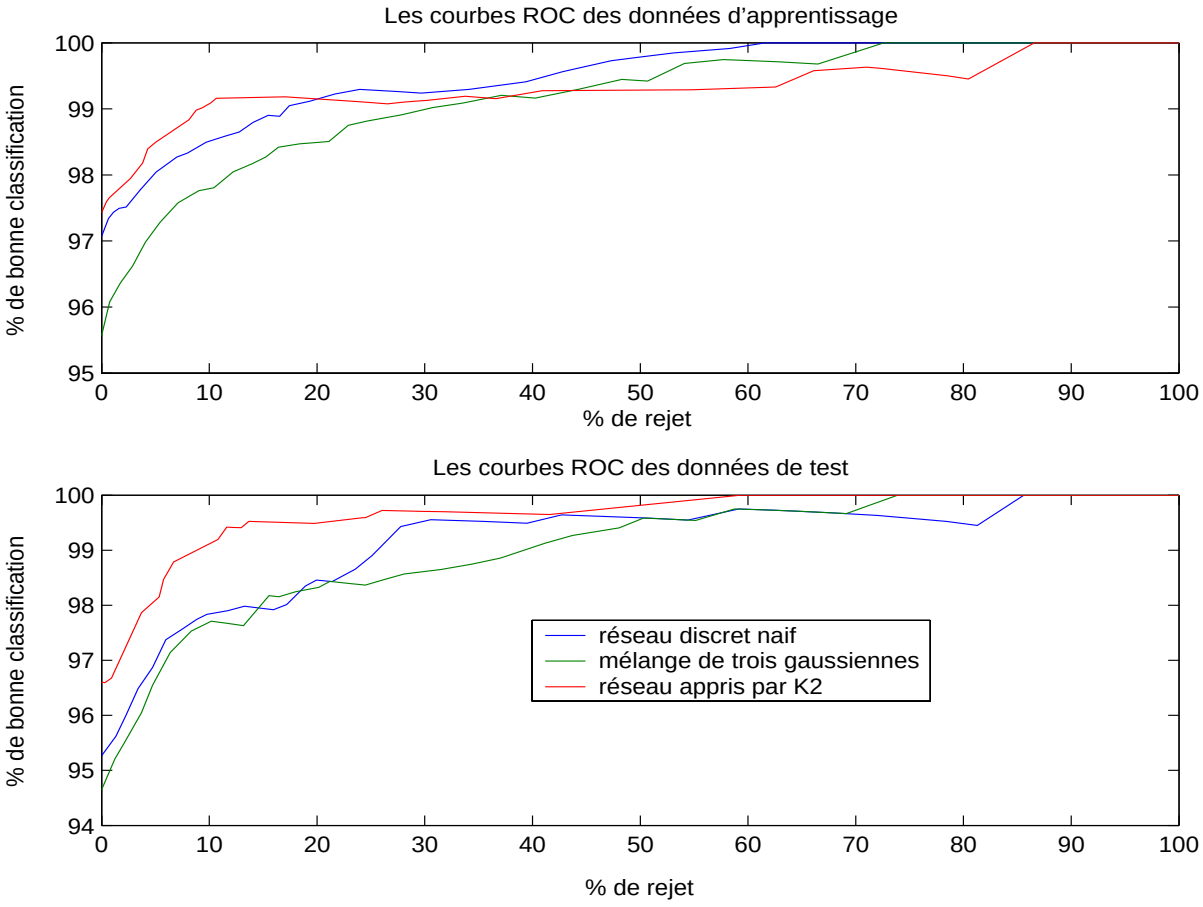


FIG. 20 – Résultats issus d'un long apprentissage.

On remarque que malgré l'apprentissage très long des paramètres, les trois types de réseaux testés resistent bien, et les courbes ROC des données d'apprentissage ne sont pas exagérément supérieures aux courbes de test, et à ma grande surprise, la courbe

d'apprentissage.

Par contre, on voit que le réseau naïf donne un avantage à la courbe d'apprentissage, sans que celui-ci ne soit trop important.

J'en conclurait donc que ces réseaux sont assez robustes, et que le fait d'affiner leurs paramètres avec les données d'apprentissage ne leur fait pas perdre leur généralité.

Un autre piège est de croire que l'on connait toutes les variables pertinentes pour notre diagnostic, car il se peut trés bien qu'il y ait une variable latente, or toutes les méthodes que j'ai implémentées ici ne permettent pas de trouver de telles variables, ni même des variables supplémentaires cachées.

Il est dommage que je n'ai pas encore réussi à faire tourner l'algorithme IC* car celui-ci est capable de détecter la possibilité d'existence d'une variable latente. Cependant il est intéressant de remarquer que l'algorithme K2 a permis de mettre en évidence les variables non pertinentes pour notre cas (à savoir si le patient est sous traitement antithyroïde, ou sous traitement au lithium, variables 5 et 12).

5 Conclusions et perspectives

Les techniques de base pour l'apprentissage dans les réseaux bayésiens sont pour l'instant peu nombreuses.

La structure du réseau étant connue, les méthodes statistiques (maximum de vraisemblance) ou bayésiennes (maximum à postériori) sont applicables à l'apprentissage des paramètres. Seulement, ces deux méthodes sont asymptôtiquement équivalentes, et reviennent à calculer les fréquences empiriques de l'observation des états des différentes variables étant donné l'état de leurs parents.

Les structures *'naïves'* permettent de gagner du temps au niveau de la génération du réseau et permettent d'obtenir des résultats convainquant. Il sera donc intéressant d'implémenter la méthode du 'Ou-Bruité' par la suite (cf [ONI00]).

Pour les problèmes d'apprentissage de la structure, les méthodes passées en revue sont efficaces tant que le nombre de variables reste faible, et les données d'apprentissages complètes.

Les algorithmes causaux sont proches de ce que l'on peut attendre de l'interprétation de la structure d'un réseau bayésien comme ensemble de dépendances causales. La mise en œuvre des algorithmes IC* et PC a été faite avec l'utilisation d'un oracle, qui donnait les valeurs des dépendances causales du graphe d'arrivée. Il faudra les faire tourner, soit avec un test du χ^2, soit avec un test du G^2, pour pouvoir comparer leur efficacité face aux autres méthodes.

L'algorithme K2 s'est avéré être le plus rapide et le plus performant des algorithmes testés sur la base de données *Thyroid*. Il semble que l'implémentation d'un algorithme d'apprentissage de la structure dans l'ensemble des équivalent de Markov puisse permettre d'obtenir de meilleurs résultats.

Il éxiste un algorithme de recherche de la structure qui peut fonctionner lorsque la base de données est incomplète. Il s'agit de l'algorithme *Structural-EM* ([FRI97], [FRI98]), qui a un principe de fonctionnement proche de l'algorithme EM. Cette méthode récente risque de devenir une référence pour l'apprentissage de la structure comme son homologue l'est pour l'apprentissage des paramètres.

Il peut être envisagé de lier plus fortement les réseaux de neurones avec les réseaux bayésiens, car on peut tout à fait admettre que la distribution de probabilité d'un nœud soit 'stockée par un réseau de neurones. On pourrait alors profiter des avantages de ces deux modèles pour faire des systèmes experts encore plus efficaces.

riationnelles, permettent de faire de l'inférence dans les réseaux bayésiens dynamiques et dans les réseaux bayésiens temporels. Aucun des ces deux modèles de réseaux n'a été développé ici. Or pour de nombreux problèmes d'optimisation, le temps, ou le résultat d'une action, peut avoir des répercutions sur l'ensemble des dépendances causales. Ces types de réseaux sont donc intéréssants, mais comme ils possèdent un grand nombre de nœuds et souvent des cliques de grande taille, il devient difficile d'y faire de l'apprentissage de la structure et même parfois de l'inférence.

On pourrait alors ajouter de la connaissance à priori, venant d'un expert par exemple, pour faire de la recherche de structure dans un espace plus restreint, et tels que les structures plus pertinentes aient une meilleur probabilité à priori.

On pourrait également envisager de faire de la moyenne de modèles (*model averaging*) : pour un problème de classification, on utilise plusieurs modèles graphiques ayant des performances moyennes. Puis on compare leurs résultats pour obtenir la bonne classification.

L'utilisation de la méthode du '*boosting*' [SCH02] permettrait d'augmenter les performances d'un algorithme de classification. Son principe est simple et consiste en l'adaptation de la base de données (en faisant varier les fréquences relative de ces différents éléments) pour que notre modèle obtienne de meilleurs taux de classification.

Le formalisme des réseaux bayésiens étant fort récent (Pearl 1988), la théorie n'en n'est qu'à ses balbutiements et les perspectives que laissent entrevoir les réseaux bayésiens dans le domaine de l'intelligence artificielle sont très vastes.

[AKA73] H. Akaike (1973) : *Information theory and extension of the maximum likelihood principle*, in Proc. of the 2nd Int. Symp. of Info. Theory, p. 267-281, Budapest.

[BOU94] R.R. Bouckaert (1994) : *Probabilist network construction using the Minimum Description Length principle*, Departement of computer science, Utrech university, Netherlands.

[BUN94] Buntine et Wray (1994) : *Operations for learning with graphical models*, JAIR 2, p. 159-225.

[BUN96] Buntine et Wray (1996) : *A guide to the litterature on learning probabilistic networks from data*, IEEE Transactions on Knowledge and data engineering, Vol. 8, No. 2, p. 195-210.

[CHI96] D.M. Chickering (1995) : *A transformation characterization of bayesian network structures*, In S. Hanks et P. Besnard editors, Proceedings of the eleventh conference on uncertainty in artificial intelligence, p. 87-98. Morgan Kaufmann.

[CHI96] D.M. Chickering (1996) : *Learning bayesian network is NP-complete*, In Learning from data : artificial intelligence and statistics V, p. 121-130. Springer-Verlag, New-York.

[CHI02] D.M. Chickering (2002) : *Learning equivalence classes of bayesian-network structures*, Journal of machine learning research 2, p. 445-498, microsoft research, Redmond, USA.

[COO92] G.F. Cooper et E. Hersovits (1992) : *A bayesian method for the induction of probabilistic networks from data*, Machine Learning 9, p. 309-347.

[DEM77] A. Dempster, N. Laird et D. Rubin (1977) : *Maximun likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society B 39, p. 1-38.

[DIE01] F.J. Diez, J. Mira, E.Iturralde, S.Zubillaga : *DIAVAL, a bayesian expert system for echocardiography*, Dept. of Atificial Intelligence. UNED., Madrid, Spain.

[FRI97] N. Friedman (1997) : *Learning belief networks in the presence of missing values and hidden variables*, in Proceedings of the fourtteenth international conference on machine learning, Morgan Kauffman, San Francisco.

[FRI98] N. Friedman (1998) : *The bayesian structural EM algorithm*, in Proceedings of the forteenth conference on uncertainty in artificial intelligence, p. 129-138, San Francisco, Morgan Kaufmann.

[FRI99] N. Friedman, X. Boyen et D. Koller (1999) : *Discovering the hidden structure of complex dynamic system*, Proceedings of the fifteenth annual conference on uncertainty in artificial intelligence (UAI-99), Stockholm, Sweden.

[GIL01] S.B. Gillispie et M.D. Perlman (2001) : *Enumerating markov equivalence classes of acyclic digraph models*, in Proceedings of the seventeenth conference on uncertainty in artificial intelligence, p. 171-177, San francisco. Morgan Kaufmann.

[GIL96] W.R. Gilks, S. Richardson et D.J. Spiegehalter (1996) : *Markov chain Monte-Carlo in pratice*, Chapman ant Hall.

[HEC94] D. Heckerman, D. Geiger et M. Chickering (1994) : *Learning networks : the combination of knoledge and statistical data*, in Proceedings of the tenth conference on uncertainty in artificial intelligence, p. 293-301, San francisco. Morgan Kaufmann.

[HEC96] D. Heckerman (1996) : *A tutorial on Learning in Bayesian Networks*, Technical Report MSR-TR-95-06, Microsoft corporation, Redmond, USA.

[HEC97] D. Heckerman (1997) : *Bayesian networks for Data-Mining*, Data Mining and Knowledge Discovery 1, Microsoft Recherch, 9S, Redmond, WA 98052-6399

[JEN90] F.V. Jensen, S.L. Lauritzen et K.G. Olesen (1990) : *Bayesian updating in causal probabilistic networks by local computations*, Computationnal Statistics quarterly, 4, p. 269-282.

[JEN96] F.V. Jensen (1996) : *Introduction to Bayesian Networks*, Springer Verlag.

[JOR98] M.I. Jordan (1998) : *Learning in Graphical Models*, Kluwer Academic Publishers.

engine, IEEE Trans. on Systems, Man and Cybernetics, 17, p. 120-132.

[KRA98] P.J. Krause (1998) : *Learning Probabilistic Networks*, Philips Research Laboratories, Crossoak Lane, Redhill, Surrey RH1 5HA, United Kingdom.

[LAU88] S.L. Lauritzen et D.J. Spiegehalter (1988) : *Local computations with probabilities on graphical structures and their applications to expert systems*, Journal of the Royal Statistical Society.

[LAU96] S.L. Lauritzen (1996) : *Graphical models*, Oxford statistical sciences series. Clarendon press.

[LAU99] S.L. Lauritzen, D.J. Spiegelhalter, R.G. Cowell et A.P. Dawid (1999) : *Probabilistic networks and expert systems*, Statistics for engineering and information science, Spinger-Verlag, New-York.

[LER02] P. Leray et O. François (2002) : *Réseaux Bayésiens et Diagnostic Médical*, soumis à la Revue d'Intelligence Artificielle.

[MAH02] P. Mahé (2002) : *Mise en œuvre d'un réseau bayésien dans une application d'aide au diagnostic médical*, Rapport d'u.v. libre, *dep*^{t} asi, insa de Rouen.

[MAT00] F. El-Matouat, O.Colot, P.Vannoorenberghe and J. Labiche (2000) : *From continous to discrete variables for bayésian network classifiers*, conférence on systems, Man and Cybernetics, IEEE-SMC, Nashville, USA.

[MUR00] K. Murphy (2000) : *The Bayes Net Toolbox 3.0 for Matlab 5.* http ://www.cs.berkeley.edu/ murphyk/Bayes/bnt.html

[NAI99] P. Naïm et A. Becker (1999) : *Les réseaux bayésiens*, Eyrolles, Paris.

[ONI98] A. Onisko, M.J. Druzdzel and H. Wasyluk (1998) : *A probabilistic causal model for diagnosis of liver disorder*, Intelligent Information System VII, Procedings of the Workshop held in Malbork, Poland

[ONI00] A. Onisko, M.J. Druzdzel and H. Wasyluk (2000) : *Learning bayesian network parameters from small data sets : application of Noisy-OR gates*, 12th European Conférence on Artificial Intelligence (ECAI2000), Berlin, Germany.

[PEA90] T. Verma et J. Pearl (1990) : *Equivalence and Synthesis of Causal Models*, in Proc. Sixth Conference on Uncertainty and Artificial Intelligence, San Francisco : Morgan Kaufmann, 220-227.

[PEA91] J. Pearl et T.S. Verma (1991) : *A theory of inferred causation*, par les éditeurs J.F. Allen, R. Fikes et E. Sandewall, KR'91.

[PEA97] J. Pearl (1997) : *Graphical Models for Probabilistic and Causal Reasoning*, Cognitive Systems lab., University of California, Los Angeles, CA 90024.

[PEA00] J. Pearl et T.S. Verma (1991) : *Principles of knowledge representation and reasoning*, pages 441-452, San Mateo, Californie, Morgan Kaufmann, 2000.

[ROB77] R.W. Robinson (1977) : *Counting unlabeled acyclic digraphs*, little c.h.c., Ed., Combinatorial Mathematics V, vol.622, Lecture Notes in Mathematics, Berlin, Springer, p28-43.

[SCH02] R.E.Schapire (2002) : *The Boosting approach to machine learning : An overview*, in MSRI Workshop on nonlinear estimation and classification.

[SIE00] B. Sierra and P. Larrañaga (2000) : *Predicting the survival in malignant skin melanoma using bayesian networks automatically induced by genetic algorithms. An empirical comparaision between different approaches*, Dept. of computer science and artificial intelligence, University of the Basque Country.

[SIE01] B. Sierra, N. Serrano, P. Larrañaga, E.J. Plasencia, I. Inza, J.J. Jiménez, P. Revuelta, M.L. Mora (2001) : *Using bayesian networks in the construction of a bi-level multi-classifier. A case study using intensive care unit patients data*, Dept. of computer science and artificial intelligence, University of the Basque Country.

[SPI00] P. Spirtes, C. Gilmour et R. Scheines (2000) : *Causation, prediction, and search*, The MIT Press, 2^{nd} edition.

[WUI00] P.H. Wuillemin (2000) : *Améliorations et Implémentations d'algorithmes de propagation dans les réseaux bayésiens*, thèse de doctorat, université de Paris 6.