UNIVERSITÉ DE
SHERBROOKE

Faculté des Sciences

Apprentissage de structures de réseaux bayesiens en présence de contraintes struturelles

Auteur	*Email*	*CIP*
BA Amadou	amadou.ba@usherbrooke.ca	baxa2202
EL JABRI Chayme	Chaymae.el.jabri@usherbrooke.ca	eljc3201
LEROUX Catherine	catherine.leroux@usherbrooke.ca	lerc2802

Présenté à :
M. Félix Camirand LEMYRE

13 décembre 2020

Table des matières

1 Introduction

Les réseaux bayésiens sont des graphes acycliques dirigés (DAG de l'anglais *directed acyclic graph*). L'apprentissage de la structure de ces graphes à partir de données est un problème difficile, même si les données sont complètes : Il s'agit d'un problème NP-difficile. Dans ce rapport, on traitera l'apprentissage de la structure d'un réseau bayésien à partir de fonctions de scores, basées sur la vraissemblance avec les données, en présence de contraintes structurelles qui permettront d'éviter la surparamétrisation du réseau.

2 Définitions de base

2.1 Réseaux bayésiens

Un réseau bayésien représente une distribution de probabilité jointe d'un ensemble de variables aléatoires, qu'on assume être catégoriques. Il peut être définit par le triplet $(\mathcal{G}, \mathcal{X}, \mathcal{P})$, où $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ est un graphe acyclique dirigé (DAG), où $V_\mathcal{G}$ est un ensemble de n nœuds associés à des variables aléatoires $\mathcal{X} = \{X_1, \cdots, X_n\}$, $E_\mathcal{G}$ est un ensemble d'arcs, $\mathcal{P}$ est une collection de fonctions de masse conditionnelles $p(X_i|\Pi_i)$, Π_i dénote les parents de X_i dans le graphe qui respecte les relations existantes dans $E_\mathcal{G}$.

On représente avec des lettres majuscules comme X_i, X_j les variables aléatoires et x_i pour représenter un état générique de X_i, qui a un espace d'états $\Omega_{X_i} = \left\{x_{i_1},\ x_{i_2},\ \cdots,\ x_{i_{r_i}}\right\}$ où $r_i = |\Omega_{X_i}| \geq 2$ est le nombre fini des catégories de X_i.

En outre $r_{\Pi_i} = |\Omega_{\Pi_i}| = \Pi_{X_t \in \Pi_i} r_t$ est le nombre des instanciations possibles de l'ensemble des parents Π_i de X_i, $\theta = (\theta_{ijk})_{\forall ijk}$ est le vecteur des paramètres en entier, et les éléments $\theta_{ijk} = p(X_{ik}|\Pi_{ij})$, avec $i \in \{1,\ \cdots,\ n\}$, $j \in \{1,\ \cdots,\ r_{\Pi_i}\}$,$k \in \{1,\ \cdots,\ r_i\}$ et $\Pi_{ij} \in \Omega_{\Pi_i}$.

Étant donné un ensemble complet de données $D = \{D_1,\ \cdots,\ D_N\}$, avec N instances, ou $D_u = \mathbf{x}_u \in \Omega_X$ est une instanciation de toutes les variables, l'objectif de l'apprentissage de structure est de retrouver le DAG $\mathcal{G}$ qui maximise une fonction de score donnée : On cherche $\mathcal{G}^* = \text{argmax}_{\mathcal{G} \in \boldsymbol{\mathcal{G}}} s_\mathcal{D}(\mathcal{G})$, où $\boldsymbol{\mathcal{G}}$ est l'ensemble de tous les DAGs contenant les nœuds X, pour une fonction de score donnée $s_\mathcal{D}$.

2.2 Fonctions de score

La plupart des scores existants dans la littérature trouvent le modèle qui corresponde le mieux aux données $\mathcal{D}$ mais qui soit le plus simple possible. Les fonctions de score sont souvent décomposables en deux termes : la fonction du log-vraisemblance, et un second terme qui tient compte de la complexité du modèle, à l'aide entre autres, du nombre de paramètres nécessaires pour représenter le réseau. Ce second terme permet d'éviter la surparamétrisation du modèle. On considère certaines fonctions de score les plus connues : le critère d'information bayésien (BIC de l'anglais *Bayesian Information Criterion*) et le critère d'information d'Akaike (AIC de l'anglais *Akaike Information Criterion*). Les fonctions de score BIC et AIC diffèrent seulement au niveau du poids du terme de pénalité :

$$\text{BIC/AIC} : s_\mathcal{D}(\mathcal{G}) = \max_\theta \mathcal{L}_{\mathcal{G},\mathcal{D}}(\theta) - t(\mathcal{G}) \cdot w, \tag{1}$$

où $t(\mathcal{G}) = \sum_{i=1}^{n}(r_{\Pi_i} \cdot (r_i - 1))$ est le nombre de paramètres indépendants. Ici $w = \log(N)/2$ pour BIC et $w = 1$ pour AIC, $\mathcal{L}_{\mathcal{G},\mathcal{D}}$ est la fonction du log-vraisemblance respectant l'ensemble de données $\mathcal{D}$ et le graphe $\mathcal{G}$:

$$\mathcal{L}_{\mathcal{G},\mathcal{D}}(\theta) = \log\left(\Pi_{i=1}^{n}\Pi_{j=1}^{r_{\Pi_i}}\Pi_{k=1}^{r_i}\theta_{ijk}^{n_{ijk}}\right) \tag{2}$$

Avec n_{ijk} indique le nombre d'éléments dans $\mathcal{D}$ qui contient x_{ik} et Π_{ij}. Notant que les valeurs $(n_{ijk})_{\forall ijk}$ dépendent du graphe $\mathcal{G}$. Une propriété importante pour tous ces critères, est que leurs fonctions de scores sont décomposables. Ainsi, on peut écrire leurs fonctions de score globales comme étant une somme de fonctions de score locales à chaque noeud du graphe, soit $s(\mathcal{G}) = \sum_{i=1}^{n} s_i(\Pi_i)$, avec :

$$\text{BIC / AIC} : s_i(\Pi_i) = \max_{\theta_i}\mathcal{L}_{\Pi_i}(\theta_i) - t_i(\Pi_i) \cdot w, \tag{3}$$

où $\mathcal{L}_{\Pi_i}(\theta_i) = \sum_{j=1}^{r_{\Pi_j}}\sum_{k=1}^{r_i} n_{ijk}\log\theta_{ijk}$, et $t_i(\Pi_i) = r_{\Pi_i} \cdot (r_i - 1)$.

Cette décomposition permet donc de calculer le score global du graphe en utilisant que les scores locaux de chaque nœud.

3 Contraintes structurelles

L'apprentissage de la structure d'un réseau bayésien consiste à vouloir prédire la structure optimale d'un réseau à partir d'un réseau de base et de ses informations ou variables. Malgré de récentes améliorations algorithmiques, l'apprentissage de la structure optimale d'un réseau bayésien à partir de données est généralement irréalisable au-delà de quelques dizaines de variables. Heureusement, de nombreuses techniques permettant de réaliser des économies de calcul considérables ont vu le jour. Parmi celles-ci, on peut citer l'apprentissage de structures en présence de contraintes sur le réseau.

Dans [CJ11] les auteurs ont travaillé avec des contraintes structurelles qui précisent où dans la structure de réseaux bayésiens des arêtes peuvent ou non être incluses. Ces contraintes contribuent à réduire l'espace de recherche et sont utiles dans de nombreuses situations. En outre, nous allons montrer, dans les sections 2.1 et 2.2, des exemples de la manière dont ces contraintes peuvent être utilisées pour apprendre des structures de réseaux bayésiens à travers différents théorèmes, corollaires et lemmes.

Mais avant il nous faut partir sur la base des trois règles suivantes :

- indegree(X_j , k, op) : indique que le noeud X_j doit avoir moins de k parents lorsque $op = lf$ ou bien avoir exactement k parents lorsque $op = eq$; k étant un entier ;
- arc(X_i ,X_j) : indique que le noeud X_i doit être un parent du noeud X_j ;
- les opérateurs logiques ou et non vont servir à manipuler et former d'autres règles à partir des deux premières.

Il faut noter qu'une contrainte structurelle ne pourra être imposée que localement et ceci tant qu'elle n'implique qu'un seul noeud et ses parents. D'une part, les parents d'un noeud X_i qui violent une contrainte ne sont jamais traités ou stockés lorsqu'on s'apprête à calculer une fonction de score locale. De plus D'autre part, des contraintes telles que arc(X_1 , X_2) $\vee$ arc(X_2 , X_3) ne peuvent pas être imposées localement, car elles définissent une condition non locale (les arcs vont à des variables distinctes, à savoir X_1 et X_2). Ainsi dans ce qui suit

nous, nous supposerons que les contraintes structurelles sont imposées localement, c'est-à-dire entre un noeud X_i et ses parents.

Dans l'apprentissage de structures de réseaux bayesien, la plupart du temps, les fonctions de scores locales ont besoin d'être calculées plusieurs fois pour évaluer les graphes candidats pour finalement en choisir le meilleur parmi eux. Il est possible d'éviter de calculer plusieurs fois ces scores locales en créeant un cache pour stocker $s(\Pi_i)$ pour chaque X_i et son parent Π_i ; s étant la fonction de score. Notez que la taille de ce cache peut être exponentielle en fonction du nombre de noeuds n. En effet en retirant X_i de l'ensemble des noeuds, il existe $2^n - 1$ sous-ensembles de $\{X_i, .., X_n\}$ qui peuvent potentiellement être considérés parents de X_i ; ce qui donne $O(n.2^n.\nu)$ en temps et mémoire pour construire un cache ; avec ν est le temps nécessaire dans le pire des cas pour calculer la fonction de score locale à chaque noeud.

Dans le but réduire la taille du cache et le nombre de calculs de scores locaux, les auteurs de [CJ11] ont énoncé dans la section 4, un certain nombre de théorèmes, corollaires et lemmes dont les plus importants sont le théorème 4 et le lemme 1.

— ***Lemme 1 :***
Soit X_i un nœud de G , un DAG candidat pour un réseau bayésien où l'ensemble des parents de X_i est Π'_i. Supposons que $\Pi_i \subset \Pi'_i$ est tel que $s_i(\Pi_i) > s_i(\Pi'_i)$ (où s est un des BIC, AIC ou critères dérivé). Alors Π'_i n'est pas l'ensemble des parents de X_i dans un DAG optimal $G*$.

— ***Théorème 4 :***
Soit BIC ou AIC étant le critère de score et X_i étant un nœud avec $\Pi_i \subset \Pi'_i$ deux ensembles parents possibles tels que $t_i(\Pi'_i) + s_i(\Pi_i) > 0$. Alors Π'_i et tous les super-ensembles $\Pi'_i \subset \Pi''_i$ ne sont pas des parents optimaux.

Le théorème 4 permet de ne pas considérer certains sous-ensemles de parents sans même avoir à les inspecter. L'idée étant de vérifier les hypothèses du théorème 4 chaque fois que le score d'un ensemble de parents Π_i de X_i est sur le point d'être calculé en prenant le meilleur score de tout sous-ensemble et de voir s'il vérifie les conditions du théorème ou non. Seuls les sous-ensembles qui ont été vérifiés conformément aux contraintes structurelles peuvent être utilisés, c'est-à-dire un sous-ensemble avec un score élevé mais qui enfreint les contraintes ne peut pas être utilisé pour éliminer ses super-ensembles (en fait, il n'est un parent valide établi au départ). Chaque fois que le théorème peut être appliqué, Π_i est écarté et tous ses super-ensembles ne sont même pas inspectés ce qui permet d'arrêter le calcul des scores le plus tôt possible, ce qui réduit le nombre de calculs nécessaires pour construire et stocker le cache. Π_i est également vérifié par rapport à Lemme 1 même si ce dernier ne peut pas permettre d'éviter d'analyser les super-ensembles de Π_i.

Ainsi, l'usage du lemme 1 et du théorème 4 ainsi que tous les résultats intermédiaires ayant permis d'aboutir à ce théorème, permettent de garantir lorsqu'il sont appliqués dans le processus d'apprentissage de structure de réseaux bayésien, l'obtention d'un cache de taille minimisée même dans le cas d'un réseau de grande taille.

4 Méthodologie

Les constraintes structurelles ont permis de réduire la taille du cache C, créé à partir de tous les scores locaux de chaque noeud pour ses ensembles de parents possibles. Il faut à présent déterminer un algorithme d'optimisation afin d'obtenir le DAG avec le score maximal, qui exploite ce cache de taille réduite. Plusieurs algorithmes ont été proposés dans la littérature, voir par exemple [SM12 ; THR10 ; CJ11 ; TEK19]. Nous nous concentrerons sur l'approche proposée dans [CJ11], soit l'algorithme branch-and-bound (B&B) schématisé dans la Figure 1. Cet algorithme prend en considération les configurations de parents optimales à chaque itération.

Algorithme Branch-and-Bound (B&B) : Nous considérons une file d'attente Q où les candidats sont placés en ordre décroissant suivant leur score individuel (i.e. le premier élément a le score le plus élevé). Chaque candidat est caractérisé par un triplet $(\mathcal{G}, \mathcal{H}, s)$, où $\mathcal{G}$ est le graphe, s est le score associé au graphe, $\mathcal{H}$ est une matrice dont les composantes déterminent si les arêtes correspondantes dans $\mathcal{G}$ peuvent être conservées ou non. $C : (X_i, \Pi_i) \mapsto \mathcal{R}$ est le cache associé aux scores locaux de toutes les variables X_i et tous leurs ensembles de parents possibles Π_i, selon les contraintes structurelles.

Q est initialisé avec un seul triplet $(\mathcal{G}, \mathcal{H}, s)$ où $\mathcal{H}$ est tel que toutes les arêtes dans $\mathcal{G}$ sont possibles et $\mathcal{G}$ correspond au graphe où pour chaque noeud la configuration des parents optimale dans C a été utilisée sans forcer l'acyclicité. Tel que résumé dans la Figure 1, nous prenons un candidat dans la file d'attente (le premier ou le dernier au choix selon l'itération). Nous vérifions s'il s'agit d'un DAG en détectant la présence de cycles. S'il n'y a pas de cycles, il s'agit d'un DAG et nous comparons son score avec celui du DAG optimal actuel. Si le score est plus élevé, il devient le DAG optimal. S'il ne s'agit pas d'un DAG alors il existe au moins un cycle dans le graphe. Nous choisissons un cycle : pour chaque arête $X_i \to X_{i+1}$ dans le cycle, nous retirons seulement cette dernière et redéfinissons les parents de X_{i+1} en utilisant la configuration optimale dans C pour laquelle $X_i \to X_{i+1}$ n'en fait pas partie. Chacun des graphes disjoints résultants est ajouté à la file d'attente Q.

Nous soulignons que les sous-graphes résultants d'un graphe non-acyclique ont toujours des scores plus faibles que ce dernier, puisque celui avait la une configuration de parents optimisés à chaque noeud. De ce fait, les éléments en haut de la file réduisent la limite supérieure du score global étant donné la contrainte d'acyclicité, alors que les candidats du bas de la file améliore la limite inférieure du score globale. Les limites supérieure et inférieure du score global viennent qu'à converger vers la solution optimale. L'algorithme prend fin quand ces deux limites convergent vers le niveau de tolérance désiré ou lorsque la file est vide. Dans [CJ11] ils ont choisi de prendre un élément sur trois du bas de la file.

5 Application

Des expériences réalisées dans [CJ11] ont démontré une amélioration exponentielle de la performance possible grâce à la présence de contraintes structurelles. L'information sur les ensembles de données utilisées est résumée dans le tableau 1. Ces données proviennent d'une archive pour l'apprentissage machine, générée à l'UCI en 2007. Ils ont utilisé leur algorithme

B&B pour trouver le réseau bayésien optimal pour chacun de ces ensembles de données. Pour des fins de concision, nous nous concentrons que sur les résultats avec le score BIC, où l'amélioration est plus significative. Dans le tableau 2 nous présentons un résumé des résultats dans [CJ11] pour le score BIC seulement. Premièrement, grâce au théorème #4 ils ont été en mesure de réduire considérablement le nombre de parents moyen par noeud. Deuxièmement, cela implique que le nombre de configurations de parents totales pour le graphe est également considérablement réduit, voir exponentiellement réduit. Finalement, le nombre d'itérations de l'algorithme dépend de la taille de l'espace de recherche. Le nombre d'itérations nécessaires pour la convergence vers la solution optimale s'en voit également significativement réduit impliquant une nette amélioration de la performance. Finalement, nous avons aussi mis en évidence dans la figure 2, la réduction du nombre de configurations possibles (soit toutes les combinaisons de parents possibles pour chaque noeud) avec le théorème #4 selon le nombre de variables dans chacun des ensembles de données. Il existe une correlation claire entre la taille de l'espace de recherche et le nombre d'itérations. On constate que le théorème #4 permet de réduire exponentiellement le nombre de configurations. En réduisant le nombre de combinaisons de parents possibles pour chaque noeud, la performance s'en voit également améliorée.

6 Conclusion

[CJ11] ont démontré numériquement que la présence de contraintes structurelles permettaient d'améliorer la performance significativement, et exponentiellement dans le cas de la fonction de score BIC. Ils ont considéré des fonctions de scores qui prenaient en compte la vraisemblance du modèle avec les données et un terme de pénalité pour éviter la surparamétrisation du modèle. Ces dernières sont décomposables et peuvent être exprimées comme un somme de scores locaux à chaque noeud. Le score local contient la vraissemblance de cette variable selon ses parents. Ils ont introduit plusieurs lemmes et théorèmes dont le théorème #4 (pour le score BIC) qui permet d'éviter la surparamétrisation locale à chaque noeud et de donc garder que les configurations de parents optimales. L'amélioration de la performance pour le score BIC est plus nette puisque que réduire les configurations de parents possibles affecte de manière plus directe la fonction de vraissemblance locale. L'introdution du lemme #1 et du théorème #4 permettent de réaliser une optimization de structures de réseaux bayésiens à plusieurs noeuds sur une plus petit espace de recherche, voir exponentiellement réduit. Ils ont aussi introduit un algorithme d'optimisation, l'algorithme B&B qui prend en compte les configurations de parents optimales à chaque itération. La possibilité d'optimiser des réseaux bayésiens de grande taille ouvre la voie à plusieurs champs de recherche, tels que les diagnostiques médicaux.

7 Figures

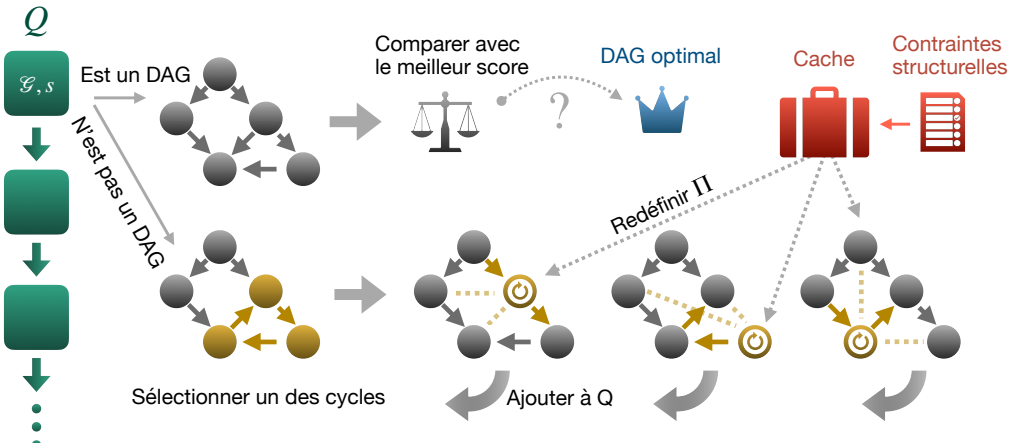

FIGURE 1 – Schéma de l'algorithme B&B tel qu'introduit dans [CJ11].

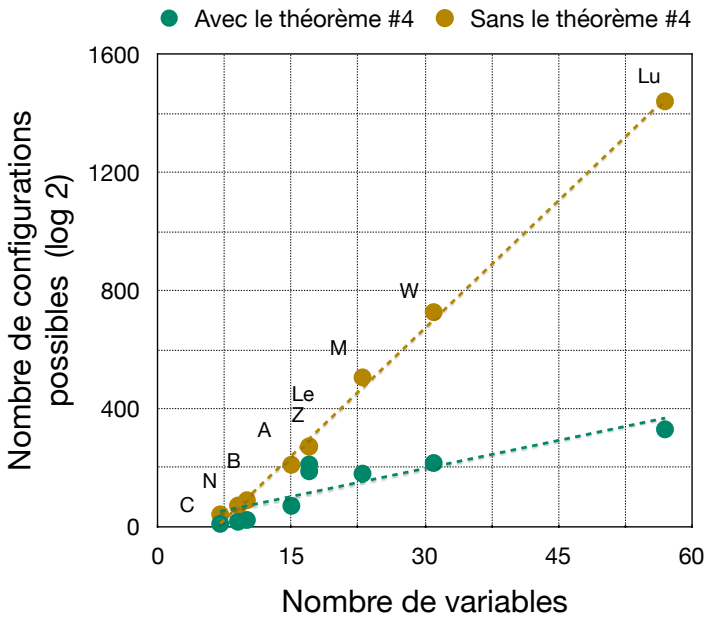

FIGURE 2 – Résultats de l'optimization avec (sans) le théorème #4 pour le score BIC. Chaque point corespond à un des ensembles de données dans le tableau 1. L'axe des x correspond au nombre de variables dans l'ensemble de données (identifié dans le graphe) et l'axe de y correspond log en base 2 du nombre de configurations possibles dans lesquelles chercher avec (sans) le théorème #4 pour le score BIC. Les courbes linéaires servent de guide.

8 Tableaux

Nom	Nombre de variables	Nombre d'échantillons
Adult (A)	15	30 162
Breast (B)	10	683
Car (C)	7	1 728
Letter (Le)	17	20 000
Lung (Lu)	57	27
Mushroom (M)	23	1 868
Nursery (N)	9	12 960
Wisconsin Diagnostic Breast Cancer (W)	31	569
Zoo (Z)	17	101

Tableau 1: Les données utilisées dans [CJ11] proviennent d'une archive de données pour l'apprentissage machine réalisée par Arthur Asuncion et David Newman en 2007 à l'université de Californie à Irvine (UCI). Les variables continues ont été discrétisées par rapport à la moyenne en variables binaires et les échantillons avec des données manquantes ont été retirés.

	A	B	C	Le	Lu	M	N	W	Z
Nombre d'itérations	$2^{14.8}$ ($2^{17.9}$)	$2^{7.3}$ ($2^{12.3}$)	$2^{8.4}$ ($2^{8.8}$)	$2^{19.0}$ ($2^{20.1}$)	$2^{15.4}$ ($2^{31.1}$)	$2^{17.1}$ ($2^{26.5}$)	$2^{10.9}$ ($2^{11.2}$)	$2^{20.7}$ ($2^{28.4}$)	$2^{13.1}$ ($2^{20.1}$)
Nombre de parents moyen par noeud	2.8 (14.0)	1.0 (9.0)	1.3 (6.0)	6.3 (16.0)	2.1 (6.0)	4.1 (22.0)	1.8 (8.0)	2.7 (8.0)	2.8 (16.0)
Nombre de configurations possibles	2^{71} (2^{210})	2^{23} (2^{90})	2^{10} (2^{42})	2^{188} (2^{272})	2^{330} (2^{1441})	2^{180} (2^{506})	2^{17} (2^{72})	2^{216} (2^{727})	2^{211} (2^{272})

Tableau 2: Résultats de l'optimization avec (sans) le théorème #4 pour le score BIC.

Références

[CJ11] Cassio P. de Campos et Qiang Ji. « Efficient Structure Learning of Bayesian Networks Using Constraints ». In : *J. Mach. Learn. Res.* 12.null (juil. 2011), p. 663-689. ISSN : 1532-4435.

[SM12] Tomi Silander et Petri Myllymäki. « A simple approach for finding the globally optimal Bayesian network structure ». In : *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)* (juin 2012).

[TEK19] Topi Talvitie, Ralf Eggeling et Mikko Koivisto. « Learning Bayesian networks with local structure, mixed variables, and exact algorithms ». In : *International Journal of Approximate Reasoning* 115 (2019), p. 69-95. ISSN : 0888-613X. DOI : https://doi.org/10.1016/j.ijar.2019.09.002. URL : http://www.sciencedirect.com/science/article/pii/S0888613X19301409.

[THR10] Jin Tian, Ru He et Lavanya Ram. « Bayesian Model Averaging Using the k-best Bayesian Network Structures. » In : jan. 2010, p. 589-597.