Approximation efficace de mélanges bootstrap d'arbres de Markov pour l'estimation de densité

François Schnitzler, Sourour Ammar, Philippe Leray, Pierre Geurts, Louis Wehenkel

Approximation efficace de mélanges bootstrap d'arbres de Markov pour l'estimation de densité

François Schnitzler[1], Sourour Ammar[2] Philippe Leray[3], Pierre Geurts[1], Louis Wehenkel[1]

[1] Université de Liège, Department of EECS et GIGA-Research, Grande Traverse, 10 - B-4000 Liège - Belgium, fschnitzler,P.Geurts,L.Wehenkel@ulg.ac.be

[2] Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes (LITIS) EA 4108, Université de Rouen, France. sourour.kessentini@univ-rouen.fr

[3] Ecole Polytechnique de l'Université de Nantes, Equipe COnnaissances et Décision, Laboratoire d'Informatique de Nantes Atlantique (LINA) UMR 6241, France. philippe.leray@univ-nantes.fr

Résumé : Nous considérons des algorithmes pour apprendre des *Mélanges bootstrap d'Arbres de Markov* pour l'estimation de densité. Pour les problèmes comportant un grand nombre de variables et peu d'observations, ces mélanges estiment généralement mieux la densité qu'un seul arbre appris au maximum de vraisemblance, mais sont plus coûteux à apprendre. C'est pourquoi nous étudions ici un algorithme pour apprendre ces modèles de manière approchée, afin d'*accélérer l'apprentissage sans sacrifier la précision*. Plus spécifiquement, nous récupérons lors du calcul d'un premier arbre de Markov les arcs qui constituent de bons candidats pour la structure, et ne considérons que ceux-ci lors de l'apprentissage des arbres suivants. Nous comparons cet algorithme à l'algorithme original de mélange, à un arbre appris au maximum de vraisemblance, à un arbre régularisé et à une autre méthode approchée.

Mots-clés : Réseaux bayésiens, arbres de Markov, mélange de modèles, bagging, randomisation, estimation de densité.

1. Introduction

L'estimation de distributions de probabilités multivariées à partir d'observations est une approche très utilisée pour traiter les problèmes de prise de décision dans un environnement incertain. Une distribution de probabilités ainsi

apprise peut fournir une réponse à des interrogations sur le problème sous-jacent, comme la probabilité d'une configuration donnée ou la distribution de probabilités d'un sous-ensemble de variables conditionnellement aux valeurs observées pour un autre sous-ensemble. Ce processus est appelé inférence.

Combinant théorie des graphes, statistique et algorithmique, le domaine des modèles probabilistes graphiques (Koller & Friedman, 2009; Pearl, 1988) fournit des outils pour modéliser des densités de probabilités et réaliser des opérations d'inférence. Parmi ces modèles, les réseaux Bayésiens encodent une distribution de probabilités conjointe sur un ensemble $\mathcal{X}$ de n variables $\{X_1, ..., X_n\}$ par le produit de la densité conditionnelle de chaque variable X_i conditionnellement à ses parents $Pa_{\mathcal{G}}(X_i)$ dans $\mathcal{G}$, un graphe orienté sans circuit dont les noeuds sont en bijection avec $\mathcal{X}$:

$$P(\mathcal{X}) = \prod_{i=1}^{n} P(X_i | Pa_{\mathcal{G}}(X_i)) \ . \tag{1}$$

Hélas, tant l'apprentissage que l'inférence est NP-difficile pour cette classe de modèles quand la structure du graphe n'est pas contrainte (Chickering *et al.*, 1994; Cooper, 1990; Kwisthout *et al.*, 2010).

Pour s'accommoder de l'augmentation rapide de la taille des problèmes rencontrés dans un grand nombre d'applications et liée à l'amélioration des méthodes d'acquisition de données, beaucoup de méthodes d'apprentissage de réseaux bayésiens imposent donc de telles contraintes, par exemple (Bach & Jordan, 2001; Friedman *et al.*, 1999b; Shahaf *et al.*, 2009). Un sous-ensemble intéressant de modèles est la classe des arbres de Markov, où chaque variable possède un unique parent, sauf la racine qui n'en a aucun (Pearl, 1988). En effet, l'apprentissage au maximum de vraisemblance par l'algorithme de Chow & Liu (1968) et l'inférence ont une complexité par rapport au nombre de variables respectivement quadratique et linéaire. Un autre avantage de ces arbres est leur faible nombre de paramètres, qui réduit le risque de surapprentissage quand il y a peu d'observations. Malheureusement, pour les problèmes définis sur un très grand nombre de variables et pour lesquels peu d'observations sont disponibles, ce nombre de paramètres peut malgré tout être encore trop élevé et il peut être avantageux d'imposer des contraintes de régularisation supplémentaires à l'algorithme de Chow-Liu (Liu *et al.*, 2011).

Le bagging (ou agrégation bootstrap) de Breiman (1996) est un méta-algorithme compensant un faible nombre d'observations par l'application répétée d'un algorithme d'apprentissage sur différents réplicats bootstrap de l'ensemble de données initial et la combinaison des prédictions des modèles

ainsi obtenus. Un réplicat bootstrap $\mathbf{D}'$ de taille p' est construit en sélectionnant de manière aléatoire, uniforme et avec remise des observations de l'ensemble initial $\mathbf{D}$ de taille p. Classiquement $p' = p$ comme c'est le cas ici. Le résultat d'un algorithme de bagging présente habituellement une variance plus faible qu'un modèle unique appris directement sur $\mathbf{D}$, et il est moins sensible aux perturbations des observations. Cette approche est assez populaire et efficace dans le contexte de l'apprentissage supervisé.

Le bagging a également été proposé pour l'estimation de densités Gaussiennes (Ormoneit & Tresp, 1995) et pour l'apprentissage de la structure de modèles probabilistes graphiques, par exemple en considérant la fréquence d'occurrence de certains éléments dans les structures obtenues sur des réplicats bootstrap, (Friedman *et al.*, 1999a), ou pour améliorer le calcul du score des différents modèles évalués en l'estimant par bootstrap (Elidan, 2011).

Dans le but d'obtenir des modèles pour lesquels l'inférence a une faible complexité, Ammar *et al.* (2009) l'ont utilisé pour générer des collections d'arbres de Chow-Liu. Cette approche est appelée *bagging d'arbres de Chow-Liu* dans la suite de ce papier, et est présentée plus en détails dans la section 3.2. Ces mélanges atteignent une meilleure précision qu'un seul arbre de Chow-Liu, en particulier sur les problèmes comportant beaucoup de variables et peu d'observations. L'apprentissage de plusieurs arbres de Chow-Liu entraîne cependant un surcoût de calcul qui peut être problématique.

C'est pourquoi nous étudions une approximation visant à diminuer ce surcoût en exploitant le calcul du premier arbre de Chow-Liu pour sélectionner un sous-ensemble d'arcs intéressants, et en ne considèrant ensuite que ceux-ci pour construire les arbres suivants. Ces arcs sont sélectionnés grâce à un test statistique évaluant l'indépendance de la paire de variables associées à chaque arc. Cette approche peut être vue comme une régularisation délimitant un squelette contenant les arcs potentiellement intéressants, et une restriction de l'espace de recherche des arbres de Markov à celui des arbres inclus au squelette. Notre approche pourrait donc également conduire à un gain en précision lorsqu'il y a peu d'observations disponibles.

Cet article est une version révisée de (Schnitzler *et al.*, 2011), améliorée par l'évaluation sur des modèles réalistes, la comparaison avec un arbre régularisé et une meilleure discussion de l'effet des paramètres.

Dans la section 2 , nous décrivons les mélanges d'arbres, et dans la section 3 quelques algorithmes pour les apprendre. Celui que nous proposons est ensuite présenté dans la section 4. Précision, vitesse de convergence et temps de calcul de tous ces algorithmes sont étudiés empiriquement dans la section 5.

2. Mélanges d'arbres de Markov

Un mélange d'arbres de Markov sur un ensemble de n variables $\mathcal{X}$ est une combinaison convexe d'une collection $\hat{\mathcal{T}} = \{T_1, \ldots, T_m\}$ de m arbres de Markov élémentaires :

$$P_{\hat{\mathcal{T}}}(\mathcal{X}) = \sum_{i=1}^{m} \mu_i P_{T_i}(\mathcal{X}) \ , \tag{2}$$

où $\{\mu_i\}_{i=1}^m$ sont les poids du mélange ($\mu_i \in [0,1]$ et $\sum_{i=1}^m \mu_i = 1$). La complexité d'inférence pour un tel mélange est donc égale à m fois la complexité d'inférence pour un arbre, qui est linéaire en le nombre n de variables.

Les algorithmes d'apprentissage de mélanges d'arbres de Markov peuvent être répartis en deux catégories : les approches au maximum de vraisemblance et de randomisation.

Dans la première, le mélange d'arbres est un moyen d'exploiter les bonnes propriétés algorithmiques des arbres tout en renforçant leurs capacités de modélisation (Meila & Jordan, 2001; Kumar & Koller, 2009).

La seconde approche peut être vue comme une approximation de l'apprentissage bayésien dans l'espace des arbres de Markov. Dans ces méthodes, les modèles d'arbre sont générés par une procédure plus ou moins aléatoire, allant de la génération de structures complètement aléatoires au mélange bootstrap d'arbres. Les poids associés à ces arbres peuvent être uniformes ou proportionnels au score de ces modèles par rapport aux observations, mais les premiers sont généralement préférables (Ammar *et al.*, 2009). Le présent article se situe dans ce contexte, et plusieurs de ces méthodes sont décrites dans la section 3. Par ailleurs, Ammar *et al.* (2010a) ont réalisé une comparaison des approches de base qui servent de référence pour notre étude.

3. Algorithmes de référence

Nous décrivons ici trois méthodes existantes pour l'estimation de distribution à l'aide d'arbres de Markov, en particulier leur complexité algorithmique. Ces méthodes seront utilisées comme point de comparaison.

3.1. L'algorithme de Chow-Liu pour apprendre un arbre de Markov

Chow & Liu (1968) ont développé l'algorithme permettant l'apprentissage de la structure d'un arbre de Markov $T_{CL}(\mathbf{D})$ maximisant la vraisemblance

de l'ensemble d'apprentissage $\mathbf{D}$. La structure de cet arbre est la solution du problème d'optimisation

$$T_{CL}(\mathbf{D}) = \arg\max_{T} \sum_{(X_{i_1}, X_{i_2}) \in \mathcal{E}(T)} I_{\mathbf{D}}(X_{i_1}; X_{i_2}) \ , \tag{3}$$

où $\mathcal{E}(T)$ désigne les arcs de T (graphe contraint à une structure d'arbre) et où $I_{\mathbf{D}}(X_{i_1}, X_{i_2})$ est l'estimation au maximum de vraisemblance basée sur $\mathbf{D}$ de l'information mutuelle entre les variables X_{i_1} and X_{i_2}.

L'algorithme de Chow-Liu (algorithme 1) contient donc deux étapes : les estimations $I_{\mathbf{D}}(X_{i_1}, X_{i_2})$ $(\forall i_1 = 1 \dots n, \forall i_2 = i_1 + 1 \dots n)$ sont d'abord calculées, remplissant une matrice symétrique $n \times n$ (MI), utilisée ensuite pour calculer un arbre de recouvrement de poids maximal (MWST).

L'étape 1 demande $\mathcal{O}(n^2 p)$ opérations, alors que calculer le MWST a une complexité $\mathcal{O}(E \log(E))$, où E est le nombre d'arcs candidats. Ici $E = n(n-1)/2$ ce qui veut dire que pour un nombre d'observations p donné, la complexité totale est $\mathcal{O}(n^2 \log(n^2)) \equiv \mathcal{O}(n^2 \log(n))$ en fonction du nombre n de variables.

Algorithme 1 (Chow-Liu (CL))

1. $MI = [0]_{n \times n}$
2. *Repeat for* $i_1 = 1, \cdots, n$ *:*
   Repeat for $i_2 = i_1 + 1, \cdots, n$ *:*
   $MI[i_1, i_2] = MI[i_2, i_1] = I_{\mathbf{D}}(X_{i_1}; X_{i_2})$
3. $T_{CL} = \mathbf{MWST}(MI)$
4. *Return* T_{CL}.

3.2. Bagging d'arbres de Chow-Liu

Cette méthode est une simple application du bagging à l'algorithme de Chow-Liu et est décrite dans l'algorithme 2. Sa complexité est m fois celle de l'algorithme de Chow-Liu, c'est-à-dire $\mathcal{O}(mn^2 \log(n))$, pour un nombre d'observations fixé. Comme expliqué précédemment, le bagging réduit la variance du modèle, ce qui augmente généralement la précision.

Par ailleurs, Schnitzler *et al.* (2010) ont montré qu'apprendre les paramètres de chaque arbre sur $\mathbf{D}$ plutôt que sur le réplicat $\mathbf{D}'$ utilisé pour apprendre sa structure améliore la précision. Nous utiliserons donc cette méthode. L'apprentissage des paramètres du modèle se réduisant au calcul de fréquences observées, nous n'en parlerons pas plus longtemps.

Algorithme 2 (Bagging d'arbres de Chow-Liu)

1. $\hat{\mathcal{T}} = \emptyset$
2. *Repeat for* $j = 1, \cdots, m$ *:*
   a) $\mathbf{D}'$ = *bootstrap(*$\mathbf{D}$*)*
   b) $\hat{\mathcal{T}} = \hat{\mathcal{T}} \cup \{\textbf{Chow-Liu}(\mathbf{D}')\}$
3. *Return* $\hat{\mathcal{T}}$, $\mu = \{1/m, \cdots, 1/m\}$.

3.3. Heuristique inertielle

Cet algorithme (Ammar *et al.*, 2010b) réduit la complexité de calcul de la méthode de bagging en limitant à K le nombre de paires de variables et d'informations mutuelles calculées et considérées pour la construction de chaque arbre. $\hat{\mathcal{T}}$ est ici construit séquentiellement : pour le premier arbre, un ensemble $\mathcal{S}$ de K arcs sélectionnés aléatoirement est considéré, et pour chaque arbre T_i après le premier, l'ensemble d'arcs est constitué des arcs de l'arbre précédent $\mathcal{E}(T_{i-1})$ et de $K - |\mathcal{E}(T_{i-1})|$ autres arcs sélectionnés aléatoirement.

Le paramètre K contrôle la complexité algorithmique de la méthode, qui est $\mathcal{O}(mK \log K)$. Pour nos comparaisons, nous utilisons $K = n \ln n$ comme suggéré par les auteurs de la méthode, ce qui donne une complexité approximativement $\mathcal{O}(mn \log(n))$.

Algorithme 3 (Heuristique inertielle pour mélange d'arbres de Markov (ISH))

1. $\hat{\mathcal{T}} = \emptyset, \mathcal{S} = \emptyset$
2. *Repeat for* $j = 1, \cdots, m$ *:*
   a) $\mathbf{D}'$ = *bootstrap(*$\mathbf{D}$*)*
   b) $MI = [0]_{n \times n}$
   c) *Repeat for* $k = 1, \cdots, |\mathcal{S}|$ *:*
      i. $(i_1, i_2) = GetIndices(\mathcal{S}[k])$
      ii. $MI[i_1, i_2] = MI[i_2, i_1] = I_{\mathbf{D}'}(X_{i_1}; X_{i_2})$
   d) *Repeat for* $k = |\mathcal{E}| + 1, \cdots, K$
      i. $(i_1, i_2) = drawNewRandomEdge$
      ii. $MI[i_1, i_2] = MI[i_2, i_1] = I_{\mathbf{D}'}(X_{i_1}; X_{i_2})$
   e) T= **MWST***(MI)*
   f) $\mathcal{S} = \mathcal{E}(T)$
   g) $\hat{\mathcal{T}} = \hat{\mathcal{T}} \cup \{T\}$
3. *Return* $\hat{\mathcal{T}}$, $\mu = \{1/m, \cdots, 1/m\}$.

4. Mélanges élagués par bagging d'arbres de Chow-Liu

Dans cette section nous décrivons notre méthode alternative pour approximer le bagging d'arbres de Chow-Liu. Tant l'arbre de Chow-Liu que le bagging de celui-ci (algorithme 1 et 2) construisent des structures d'arbres connexes maximisant la vraisemblance des données d'apprentissage en considérant tous les arcs. L'approximation ISH (algorithme 3) en utilise un nombre plus faible pour construire chaque arbre, et améliore itérativement les arcs considérés en combinant exploration et réutilisation des arcs composant l'arbre précédent. Nous jugeons plus efficace d'explorer tous les arcs et d'en identifier un sous-ensemble intéressant lors de la construction du premier arbre, puis de ne considérer que ces arcs lors de l'apprentissage des arbres suivants.

Dans les problèmes en très grande dimension ($p \ll n$), maximiser la vraisemblance des données sur tous les arbres connexes peut déjà entrainer du surapprentissage. Il est donc intéressant de considérer une régularisation structurelle supplémentaire de l'algorithme de Chow-Liu, en modifiant le critère d'optimisation (équation 3), de manière à pénaliser sa complexité en terme du nombre d'arcs $|T|$:

$$T_{CL}^{\lambda}(\mathbf{D}) = \arg\max_{T} \sum_{(X_{i_1}, X_{i_2}) \in \mathcal{E}(T)} I_{\mathbf{D}}(X_{i_1}; X_{i_2}) - \lambda |T| \ , \qquad (4)$$

où T peut maintenant être une forêt de Markov (*au maximum* un chemin entre chaque noeud). La solution à ce problème est fournie par un algorithme de Chow-Liu modifié pour renvoyer la forêt obtenue en enlevant de l'arbre optimal ($\lambda = 0$) les arcs associés à une information mutuelle $I_{\mathbf{D}}(X_{i_1}; X_{i_2})$ inférieure à λ.

Comme pour la construction d'arbres de décision (Wehenkel, 1993), nous constatons que pénaliser de cette façon la complexité de l'arbre équivaut à utiliser un test d'hypothèse pour vérifier l'indépendance des arcs de l'arbre. Un tel test peut être formulé en comparant la quantité $2p(\ln 2) I_{\mathbf{D}}(X_{i_1}; X_{i_2})$ (distribuée, en cas d'indépendance de X_{i_1} et X_{i_2}, selon une loi χ^2 à $(|X_{i_1}| - 1)(|X_{i_2}| - 1)$ degrés de liberté) à une valeur seuil correspondant à un risque de première espèce choisi, par exemple $\alpha = 0.05$. Donc, l'arc associé à une paire (X_{i_1}, X_{i_2}) ne sera pas inclus dans la forêt construite par l'algorithme de Chow-Liu modifié quand $2p(\ln 2) I_{\mathbf{D}}(X_{i_1}; X_{i_2})$, estimé à partir des observations, est inférieur à une valeur seuil calculée à partir de la loi χ^2 et α.

Dans un contexte de mélange, les calculs effectués pour construire le premier arbre par l'algorithme de Chow-Liu sont utilisés pour identifier ces paires

de variables pour lesquelles l'information mutuelle est au-dessus du seuil. Nous considérons ensuite uniquement l'ensemble $\mathcal{S}$ des arcs associés à ces paires de variables pour construire les arbres composant le reste du mélange (Algorithme 4) : $\mathcal{S}$ délimite un espace de structures restreint dans lequel nous recherchons celle qui maximise la vraisemblance de chaque réplicat bootstrap. La diversité des structures d'arbre est liée à la taille de $\mathcal{S}$. L'identification d'un espace réduit puis l'utilisation de méthodes plus poussées pour affiner la recherche est également utilisée par certaines méthodes d'apprentissage de structure unique, par exemple (Friedman *et al.*, 1999b; Aliferis *et al.*, 2010).

Algorithme 4 (Pré-élagage pour le bagging d'arbres de Chow-Liu (PMBCL))

1. $\mathcal{S} = \emptyset$, $MI = [0]_{n \times n}$
2. *Repeat for* $i_1, i_2 > i_1$; $i_1, i_2 \in [1, \cdots, n]$
   if $I_{\mathbf{D}}(X_{i_1}; X_{i_2}) > \lambda(\alpha)$
   a) $MI[i_1, i_2] = MI[i_2, i_1] = I_{\mathbf{D}}(X_{i_1}; X_{i_2})$
   b) $\mathcal{S} = \mathcal{S} \cup (i_1, i_2)$
3. $\hat{\mathcal{T}} = \mathbf{MWST}(MI)$
4. *Repeat for* $j = 2, \cdots, m$:
   a) $\mathbf{D}'$ = *bootstrap*($\mathbf{D}$)
   b) *Repeat for* $k = 1, \cdots, |\mathcal{S}|$:
      i. $(i_1, i_2) = GetIndices(\mathcal{S}[k])$
      ii. $MI[i_1, i_2] = MI[i_2, i_1] = I_{\mathbf{D}'}(X_{i_1}; X_{i_2})$
   c) $\hat{\mathcal{T}} = \hat{\mathcal{T}} \cup \mathbf{MWST}(MI)$
5. *Return* $\hat{\mathcal{T}}$, $\mu = \{1/m, \cdots, 1/m\}$.

La complexité de l'algorithme 4 est $\mathcal{O}(n^2 + mK(\alpha)\log(K(\alpha)))$: le terme quadratique ne dépend plus de la taille du mélange et le second, similaire à celui d'ISH, dépend du nombre d'arcs candidats $K(\alpha) \equiv |\mathcal{S}|$, et donc du paramètre α (comme $|\mathcal{S}|$ croit de 0 à $n(n-1)/2$ lorsque α croit de 0 à 1).

5. Validation expérimentale

Nous comparons empiriquement notre algorithme de la section 4 à ceux de la section 3. Pour ce faire nous utilisons d'abord des distributions cibles simulées et encodées par un réseau bayésien sur des variables binaires, puis des distributions plus réalistes.

Chaque **distribution simulée** est construite en considérant les variables séquentiellement et en sélectionnant aléatoirement pour chacune un nombre de parents dans $[0, max(5, i-1)]$ et ces parents dans $\{X_1, ..., X_{i-1}\}$. Les paramètres sont échantillonnés à partir d'une loi de Dirichlet uniforme. Les résultats portent sur 200 et 1000 variables et $p = 200, 600, 1000$ observations, soit peu d'observations par rapport au nombre de variables. Pour chaque configuration, les résultats sont moyennés sur 5 distributions cibles fois 6 ensembles d'apprentissage.

La qualité de l'estimation de la distribution en terme de précision est mesurée par une estimation Monte-Carlo de la divergence de Kullback-Leibler (Kullback & Leibler, 1951), une mesure asymétrique de la dissimilarité d'une distribution donnée $P_{\hat{\mathcal{T}}}$ par rapport à une distribution cible P :

$$\hat{D}_{KL}(P \mid\mid P_{\hat{\mathcal{T}}}) = \frac{1}{N} \sum_{X \sim P}^{N} \log_2 \left(\frac{P(X)}{P_{\hat{\mathcal{T}}}(X)} \right) \quad . \tag{5}$$

Nous utilisons le même ensemble indépendant de $N = 50000$ observations pour toutes les évaluations correspondant à la même distribution cible.

Sur chaque ensemble d'apprentissage, nous appliquons les quatre algorithmes présentés pour un nombre croissant de termes (m=1,10,20...), sauf pour l'algorithme de Chow-Liu qui ne produit qu'un arbre. Nous évaluons ainsi la précision, le temps de calcul et la vitesse de convergence des différentes méthodes et l'impact du nombre d'observations.

Les **distributions réalistes** sont celles fournies dans le supplément web de Aliferis *et al.* (2010). Outre étudier l'intérêt des mélanges sur des problèmes plus proches de la réalité, ces expériences permettent de mieux illustrer l'intérêt des mélanges par rapport à un arbre régularisé. La qualité des modèles est pour ces distributions évaluée par la log-vraisemblance négative d'un ensemble test de 5000 observations.

5.1. Evaluation de la précision des modèles

Commençons par évaluer les performances relatives des différents algorithmes en terme de précision pour 1000 variables et 200 observations. La figure 1 affiche la divergence de Kullback-Leibler par rapport à la distribution cible pour un arbre de Chow-Liu (CL, Algorithme 1) et pour les autres méthodes en fonction de la taille du mélange m. La méthode proposée (PMBCL, algorithm 4) est testée ici avec 4 valeurs différentes de son paramètre α : $10^{-1}, 5E^{-2}, 5E^{-3}, 5E^{-4}$.

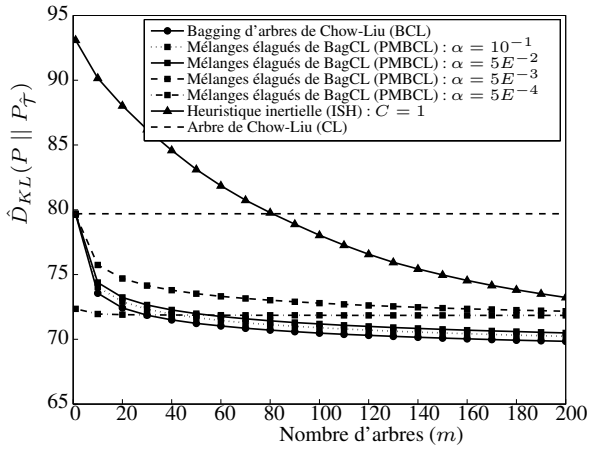

FIGURE 1: Précision des méthodes présentées (pour $n = 1000$ et $p = 200$). Abscisse : nombre d'arbres m ; ordonnée : valeur moyenne de la divergence KL (estimée par Monte-Carlo) sur 5 distributions cibles simulées $\times$ 6 ensembles de données $\mathbf{D}$.

Au niveau du premier arbre de chaque mélange ($m = 1$), toutes les méthodes sauf deux débutent au même point que l'arbre de Chow-Liu : ISH (Algorithme 3) est bien pire tandis que la méthode PMBCL la plus régularisée ($\alpha = 5E^{-4}$) est significativement meilleure [1]. Lorsque m augmente, tous les mélanges s'améliorent monotonément, certains plus vite que d'autres. Pour des valeurs de m suffisamment grandes elles sont toutes plus précises que l'arbre de Chow-Liu. Pour PMBCL, un α plus petit diminue le gain obtenu en augmentant m. Pour $\alpha = 5E^{-4}$, la vitesse d'amélioration est si faible que sa performance est très vite dépassée par BCL (pour $m = 30$) et plus tard par PMBCL avec $\alpha = 5E^{-2}$ (pour $m = 60$). Par ailleurs, PMBCL avec α assez grand affiche une précision et une vitesse de convergence semblable à BCL.

Pour expliquer l'influence de α, nous montrons à la Table 1 le nombre d'arcs candidats $|S|$ (en absolu et en % par rapport à la valeur maximale $n(n-1)/2 = 499500$) et le nombre $|T_1|$ d'arcs retenus dans le premier terme du mélange PMBCL. Nous voyons que $|T_1|$ reste quasi maximal ($|T_1| \simeq n - 1 = 999$) pour $\alpha \in \{10^{-1}, 5E^{-2}, 5E^{-3}\}$, méthodes dont les courbes ont initialement la même précision que CL, tandis que le plus faible α ($5E^{-4}$) conduit à $|\mathcal{S}| \simeq 0.68(n-1)$ et à $|T_1| \simeq 0.63(n-1)$.

1. Les intervalles de confiance ne sont pas représentés pour préserver la lisibilité des résultats, mais sont 20 fois plus petits que les écarts moyens sur lesquels nous attirons l'attention.

TABLE 1: Effet de α sur le nombre d'arcs utilisés par PMBCL. Moyenne sur 5 distributions $\times$ 6 ensembles de données ; $n = 1000$ et $p = 200$.

	Nombre d'arcs (% du total) pour $\alpha =$			
	10^{-1}	$5E^{-2}$	$5E^{-3}$	$5E^{-4}$
$\|T_1\|$	998	997.9	993.2	626.8
$\|S\|$	52278(10.5%)	26821(5.36%)	3311(0.66%)	683 (0.13%)

Pour toutes les valeurs de α, l'inclusion d'autres modèles dans le mélange PMBCL améliore la précision. Cet effet de réduction de la variance diminue cependant avec α, et pour la valeur $\alpha = 5E^{-4}$, pour laquelle les arbres du mélange présentent une faible diversité, cela se traduit par une convergence très rapide de la précision vers une valeur proche de celle du modèle initial. La réduction de α conduit aussi à une augmentation du biais de la méthode, comme le montrent les précisions pour $m = 200$. Le comportement apparemment atypique pour α $(5E^{-4})$ s'explique par le fait que pour cette valeur la faible taille de $\mathcal{S}$ contraint fortement la structure et le nombre de paramètres des termes du modèle. La meilleure précision initiale s'explique par une réduction de variance bien plus importante que l'augmentation du biais.

5.1.1. Effet du nombre d'observations p.

Pour analyser plus finement le comportement des différentes méthodes, p est augmenté à 1000 observations. Le changement le plus visible dans les résultats rapportés dans la figure 2 concerne le point de départ des différentes méthodes ($m = 1$). CL est au départ plus intéressant qu'un arbre construit sur un réplicat bootstrap et l'avantage du premier arbre régularisé de PMBCL diminue, deux conséquences logiques de la meilleure estimation des informations mutuelles grâce aux observations supplémentaires.

Une meilleure estimation des "bons" arcs semble inverser l'effet d'une diminution d'α. Il faut noter que, même si pour $m = 100$, la plus faible valeur prise par α semble plus intéressante en moyenne, les intervalles de confiance (non-affichés) suggèrent que les méthodes ne peuvent être discriminées.

5.2. Temps de calcul

Nos expériences ont été réalisées sur une grille de calcul sous ClusterVisionOS composée de paires de processeurs Intel L5420 2.50 Ghz. Etant donné

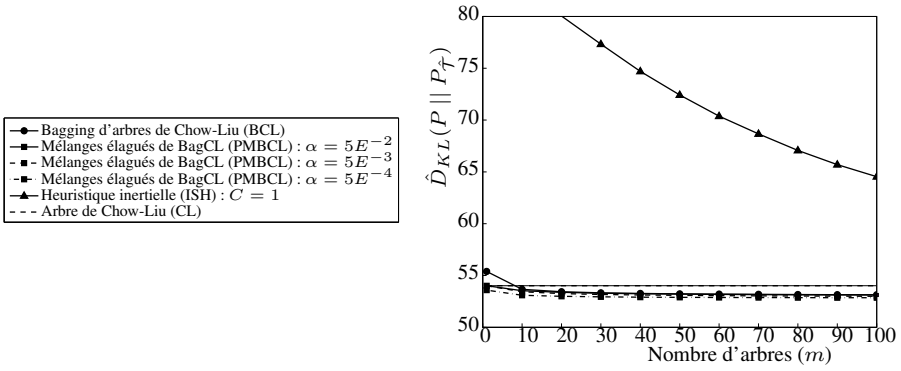

FIGURE 2: Précision des méthodes présentées (pour $n = 1000$ et $p = 1000$). Abscisse : nombre d'arbres m ; ordonnée : valeur moyenne de la divergence KL (estimée par Monte-Carlo) sur 5 distributions cibles simulées $\times$ 6 ensembles de données $\mathbf{D}$.

TABLE 2: Temps de calcul série minimum

Méthode	temps relatif minimum pour l'apprentissage					
	n=200, m=500 - sauf CL			n=1000, m=100 - sauf CL		
	p=200	p=600	p=1000	p=200	p=600	p=1000
CL	1	3.07	5.3	37	98	174
BCL	532	1531	2674	5037	11662	19431
ISH	45	186	432	181	800	1433
PMBCL	21	82	191	139	612	1005

l'environnement partagé, le temps de calcul d'une même méthode peut varier énormément. Nous avons donc choisi d'indiquer ici le temps d'apprentissage minimum constaté pour chaque méthode, relativement à celui de CL (pour $n = p = 200$), résultats affichés dans la table 2 pour respectivement 200 et 1000 variables / 500 et 100 arbres, avec $\alpha = 0.005$ pour PMBCL et $C = 1$ pour ISH. Considérer le temps moyen ne modifie pas les conclusions.

Ces chiffres montrent que notre approximation est environ un ordre de grandeur plus rapide que la méthode originale (BCL), avec une différence qui s'accentue lorsque le nombre de variables augmente, tout en présentant une performance et une vitesse de convergence équivalentes. Par rapport à ISH, notre méthode présente un temps d'exécution similaire pour une précision grandement améliorée.

5.3. Réseaux réalistes

Les résultats obtenus sur les réseaux réalistes sont repris dans la table 3, où le meilleur résultat pour chaque distribution et nombre d'observations est affiché en gras. Pour comparer les mélanges d'arbres à la régularisation d'un arbre de Chow-Liu, une méthode regCL, pour "Chow-Liu régularisé", vient se rajouter aux quatre méthodes vues précédemment (pour lesquelles $m = 100$). Il s'agit là du score moyen (sur 5 ensembles d'apprentissage), de la forêt de Markov dont le nombre d'arcs est optimisé pour maximiser ce score moyen. Il s'agit donc d'une régularisation parfaite pour celui-ci. Au contraire, PMBCL n'est testé qu'avec la valeur non-optimisée $\alpha = 0.05$.

La méthode regCL est donc clairement favorisée par rapport aux mélanges de modèles. Malgré son avantage, cette méthode est rarement la meilleure, ce qui montre l'intérêt des mélanges d'arbres de Markov.

Au niveau des mélanges eux-mêmes, la précision d'ISH est à nouveau assez faible, et régulièrement très mauvaise. PMBCL au contraire est souvent très proche de BCL en terme de précision. Des disparités apparaissent néanmoins dans quelques cas, et la méthode est parfois moins bonne (Alarm10 pour $p = 200$ ou Insurance10), et parfois clairement meilleure (Munin).

Cette précision supérieure pour PMBCL s'observe lors d'une importante différence entre CL et regCL, c'est-à-dire lorsque la régularisation améliore l'arbre de Chow-Liu. PMBCL présente donc deux avantages par rapport à BCL, d'une part l'accélération de la construction du mélange et d'autre part la régularisation elle-même, qui permet d'améliorer la précision du modèle.

Cette régularisation détériore parfois la précision de PMBCL (voir Insurance10), malgré l'utilisation de tests d'hypothèse. Il semble donc intéressant d'optimiser α pour chaque problème, par exemple par validation croisée.

6. Conclusion

Nous nous intéressons à la combinaison de modèles probabilistes graphiques pour l'estimation de densité et plus précisément les mélanges d'arbres de Markov pour réduire la variance. Ces modèles sont particulièrement adaptés aux problèmes définis sur un grand nombre de variables grâce au bon passage à l'échelle des algorithmes associés.

Nous proposons et étudions un algorithme pour l'apprentissage de mélanges d'arbres de Markov en vue d'approcher la qualité d'estimation du bagging d'arbres de Chow-Liu en accélérant fortement l'apprentissage.

TABLE 3: Log-vraisemblance négative des modèles appris pour estimer les réseaux réalistes (moyenne sur 5 ensembles d'apprentissage)

Distribution	n	p	CL	regCL	BCL	ISH	PMBCL
Alarm10	370	200	166.65	166.65	**163.59**	206.34	166.80
Alarm10	370	500	136.37	136.28	**135.31**	181.82	135.61
Child10	200	200	135.29	135.08	**133.94**	159.74	134
Child10	200	500	131.71	131.71	**131.01**	156.11	131.02
Gene	801	200	485.21	483.6	482.80	585.79	**482.66**
Gene	801	500	477.48	476.75	**473.75**	572.13	473.79
Hailfinder10	560	200	550.85	**547.64**	551.75	651.75	549.89
Hailfinder10	560	500	523.81	**523.26**	523.61	628.45	523.31
Insurance10	270	200	210.1	210.1	**206.77**	243.57	215.23
Insurance10	270	500	198.87	198.87	**195.47**	234.98	202.01
Lung Cancer	800	200	435.72	**435.46**	437.41	497.96	436.01
Lung Cancer	800	500	424.69	424.44	418.31	493.42	**418.30**
Munin	189	200	42.614	36.987	41.799	41.749	**35.566**
Munin	189	500	37.66	35.414	37.656	38.131	**35.140**
Pigs	441	200	390.75	390.75	**387.19**	428.55	387.24
Pigs	441	500	385.59	385.59	**382.22**	423.71	382.26

Notre approche permet d'atteindre une précision proche de la méthode initiale pour un temps apprentissage réduit d'un ordre de grandeur. L'idée principale réside dans l'utilisation de l'information obtenue lors du calcul du premier arbre pour réduire le nombre d'arcs considérés lors de la construction des arbres suivants. Ces arcs correspondent uniquement aux paires de variables pour lesquelles l'information mutuelle évaluée lors du calcul du premier arbre est supérieure à un seuil lié à un test d'indépendance.

Nous avons également montré expérimentalement l'intérêt de la combinaison d'arbres de Markov par rapport à la régularisation d'un unique arbre de Chow-Liu. Ainsi, tant le bagging que notre approximation surpasse généralement la performance de ce modèle unique.

D'autres variantes de cette méthode méritent d'être étudiées dans le futur. Par exemple, il pourrait être intéressant d'effectuer une sélection moins sévère des arcs lors de la première itération, et d'inclure une étape de régularisation supplémentaire lors de l'apprentissage des termes suivants. Une telle méthode pourrait permettre de mieux fixer le compromis entre la réduction de variance potentielle et le gain en complexité.

Remerciements.

F. Schnitzler bénéficie d'une bourse F.R.I.A. de la Communauté Française de Belgique. Ce travail est également financé par le réseau IUAP Biomagnet de la Politique scientifique fédérale belge, et par le réseau d'excellence PASCAL2 de la CE. La responsabilité scientifique est celle des auteurs.

Références

ALIFERIS C., STATNIKOV A., TSAMARDINOS I., MANI S. & KOUTSOUKOS X. (2010). Local causal and Markov blanket induction for causal discovery and feature selection for classification. Part I : Algorithms and empirical evaluation. *JMLR*, **11**, 171–234.

AMMAR S., LERAY P., DEFOURNY B. & WEHENKEL L. (2009). Probability density estimation by perturbing and combining tree structured Markov networks. In *Proceedings of ECSQARU*, p. 156–167.

AMMAR S., LERAY P., SCHNITZLER F. & WEHENKEL L. (2010a). Sub-quadratic Markov tree mixture learning based on randomizations of the Chow-Liu algorithm. In *Proceedings of the 5th European Workshop on Probabilistic Graphical Models (PGM 2010)*, p. 17–24.

AMMAR S., LERAY P. & WEHENKEL L. (2010b). Sub-quadratic Markov tree mixture models for probability density estimation. In *Proceedings of the 19th International Conference on Computational Statistics*, p. 673–680.

BACH F. R. & JORDAN M. I. (2001). Thin junction trees. In *Advances in Neural Information Processing Systems 14*, p. 569–576 : MIT Press.

BREIMAN L. (1996). *Arcing Classifiers*. Technical report, Dept. of Statistics, University of California.

CHICKERING D., GEIGER D. & HECKERMAN D. (1994). Learning bayesian networks is NP-hard. *Microsoft Research*, p. 94–17.

CHOW C. & LIU C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theory*, **14**, 462–467.

COOPER G. (1990). The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, **42**(2-3), 393–405.

ELIDAN G. (2011). Bagged structure learning of bayesian network. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTAT 2011)*, p. 251–259.

FRIEDMAN N., GOLDSZMIDT M. & WYNER A. (1999a). Data analysis with bayesian networks : A bootstrap approach. In *Proceedings of the 15th*

Conference on Uncertainty in Artificial Intelligence (UAI 99), p. 196–205, San Francisco, CA, USA : Morgan Kaufmann Publishers.

FRIEDMAN N., NACHMAN I. & PEÉR D. (1999b). Learning bayesian network structure from massive datasets : The "sparse candidate" algorithm. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI 99)*, p. 206–215.

KOLLER D. & FRIEDMAN N. (2009). *Probabilistic Graphical Models*. MIT Press.

KULLBACK S. & LEIBLER R. (1951). On information and sufficiency. *ANN MATH STAT*, **22**(1), 79–86.

KUMAR M. P. & KOLLER D. (2009). Learning a small mixture of trees. In *Advances in Neural Information Processing Systems 22*, p. 1051–1059.

KWISTHOUT J. H., BODLAENDER H. L. & VAN DER GAAG L. (2010). The necessity of bounded treewidth for efficient inference in bayesian networks. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, p. 623–626.

LIU H., XU M., HAIJIE GU A. G., LAFFERTY J. & WASSERMAN L. (2011). Forest density estimation. *JMLR*, **12**, 907–951.

MEILA M. & JORDAN M. (2001). Learning with mixtures of trees. *JMLR*, **1**, 1–48.

ORMONEIT D. & TRESP V. (1995). Improved gaussian mixture density estimates using bayesian penalty terms and network averaging. In *Advances in Neural Information Processing Systems*, p. 542–548 : MIT Press.

PEARL J. (1988). *Probabilistic reasoning in intelligent systems : networks of plausible inference*. Morgan Kaufmann.

SCHNITZLER F., AMMAR S., LERAY P., GEURTS P. & WEHENKEL L. (2011). Efficiently approximating Markov tree bagging for high-dimensional density estimation. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2011), Part III*, p. 113–128.

SCHNITZLER F., LERAY P. & WEHENKEL L. (2010). Towards sub-quadratic learning of probability density models in the form of mixtures of trees. In *Proceedings of the 18th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, p. 219–224.

SHAHAF D., CHECHETKA A. & GUESTRIN C. (2009). Learning thin junction trees via graph cuts. *Artificial Intelligence and Statistics (AISTATS)*, p. 113–120.

WEHENKEL L. (1993). Decision tree pruning using an additive information quality measure. In *Uncertainty in Intelligent Systems*, p. 397–411. Elsevier - North Holland.