Mélanges sous-quadratiques d'arbres de Markov pour l'estimation de la densité de probabilité

Sourour Ammar, Philippe Leray, Louis Wehenkel

Mélanges sous-quadratiques d'arbres de Markov pour l'estimation de la densité de probabilité

Sourour Ammar* — **Philippe Leray*** — **Louis Wehenkel****

* *Equipe Connaissances et Décision (COD)*
Laboratoire d'Informatique de Nantes Atlantique (LINA) UMR 6241
Ecole Polytechnique de l'Université de Nantes, France
{sourour.ammar, philippe.leray}@univ-nantes.fr

** *Department of Electrical Engineering and Computer Science & GIGA-Research, University of Liège, Belgium*
L.Wehenkel@ulg.ac.be

RÉSUMÉ. *Afin d'explorer l'intérêt du principe "Perturber et Combiner" dans le domaine de l'estimation de densité de probabilité, nous proposons d'étudier des mélanges d'arbres de Markov inspirés du bagging utilisant un algorithme de recouvrement maximal (Chow et Liu). Nous essayons d'accélérer la procédure de recherche en réduisant sa complexité en dessous du quadratique et en essayant de conserver des performances similaires. Nous évaluons empiriquement les performances de ces heuristiques proposées du point de vue performance et complexité algorithmique et les comparons à d'autres mélanges d'arbres baggés dont la complexité est quadratique et à des mélanges d'arbres aléatoires dont la complexité est linéaire. Le résultat le plus intéressant est que nos méthodes sous-quadratiques proposées permettent d'avoir des résultats nettement meilleurs que ceux obtenus avec les méthodes aléatoires dont la complexité est linéaire et approchent la méthode de base (quadratique) lorsque le nombre de composantes du mélange augmente.*

ABSTRACT. *To explore the "Perturb and Combine" idea for estimating probability densities, we study mixtures of tree structured Markov networks derived by bagging combined with the Chow and Liu maximum weight spanning tree algorithm and we try to accelerate the research procedure by reducing its computation complexity below the quadratic and keepingg similar accuracy. We empirically assess the performances of these heuristics in terms of accuracy and computation complexity, with respect to mixtures of bagged Markov trees whose complexity is quadratic, and single random tree mixtures whose complexity is linear. We find that our pro-*

posed heuristics whose complexity is sub-quadratic outperform other random linear methods and approache the quadratic baseline method when the number of mixture components grows.

MOTS-CLÉS : *Perturber et Combiner, estimation de densité, mélanges d'arbres de Markov.*

KEYWORDS: *Perturb and Combine, density estimation, mixtures of markov trees.*

1. Introduction

L'apprentissage de structure de modèles probabilistes revient essentiellement à trouver une factorisation maximale de la densité jointe d'un ensemble de variables aléatoires, ceci en se basant sur un ensemble d'observations jointes de ces variables (Cowell *et al.*, 1999). Un tel modèle graphique probabiliste peut être utilisé pour la description d'indépendances conditionnelles dans la distribution ayant généré les données, pour le raisonnement automatique dans le domaine de l'incertain et pour des simulations de Monte-Carlo. Malheureusement, les algorithmes d'apprentissage de structure de modèles graphiques disponibles actuellement sont restrictifs du point de vue de la nature de la structure qu'il cherchent, et ils ont une complexité algorithmique importante pour être applicables pour un très grand nombre de variables. De plus, le comportement de ces méthodes dans le cas d'un nombre réduit d'individus n'est pas connu. Il est possible que ces méthodes souffrent du phénomène de sur-apprentissage lorsque le nombre de variables est très grand par rapport au nombre d'individus.

Dans le contexte de l'apprentissage supervisé, un principe générique appelé "Perturber et Combiner" a mené à beaucoup d'innovations fructueuses. Son idée principale est d'une part de perturber selon différentes directions un algorithme d'optimisation utilisé pour produire un prédicteur à partir d'un ensemble de données, et d'autre part combiner de façon appropriée un ensemble de prédicteurs obtenus par plusieurs itérations de l'algorithme perturbé sur l'ensemble de données. Dans ce principe, des ensembles de modèles aléatoires de performance faible ont été étudiés et utilisés avec succès pendant les deux dernières décennies. Parmi les avantages de ces méthodes, citons l'amélioration potentielle du passage à l'échelle de leur algorithmes d'apprentissage (e.g. (Geurts *et al.*, 2006)).

Dans le contexte de l'estimation de densité, le Bagging (et le boosting) de distributions normales a été proposé par (Ridgeway, 2002). Une description des mélanges de modèles dans le cadre de la modélisation bayésienne a été faite dans (Ammar *et al.*, 2009). Dans (Ammar *et al.*, 2008), nous avons exploré l'idée du "Perturber et Combiner" pour l'estimation de densité de probabilité avec des modèles graphiques probabilistes en comparant de grands ensembles aléatoires de poly-arbres (dirigés) et d'arbres non dirigés. Dans (Ammar *et al.*, 2009), d'autres comparaisons ont été faites essentiellement entre des ensembles d'arbres générés en utilisant l'algorithme Chow et Liu (Chow *et al.*, 1968) sur une copie bootstrap des données, de complexité quadratique en fonction du nombre de variables (appelés bagging d'arbres de Markov), et des mélanges d'arbres générés aléatoirement avec une complexité linéaire en fonction du nombre de variables. Ce travail a prouvé que les ensembles d'arbres de Markov *Baggés* sont nettement meilleurs que ceux générés aléatoirement, du point de vue performance et rapidité de convergence lorsque le nombre de composantes du mélange augmente. Ainsi, nous nous concentrons dans ce travail sur nos méthodes utilisant le bagging et étudions différentes méthodes pour améliorer ces méthodes en forçant la complexité de l'étape d'optimisation de l'algorithme *Chow Liu MWST* à descendre en dessous du quadratique tout en essayant de garder des performances similaires.

L'idée principale de ce travail est d'affaiblir la procédure de recherche de l'algorithme *Chow Liu* (MWST), qui est l'étape la plus coûteuse, en ne considérant qu'un nombre réduit de termes choisis aléatoirement dans la matrice d'information mutuelle. Nous considérons deux façons pour choisir les termes qui seront considérés dans la procédure de recherche. La première (approche naïve) consiste à choisir aléatoirement ces termes, la deuxième (approche intelligente) exploite le résultat des itérations précédentes pour choisir une partie des termes à considérer. Nous comparons les performances de ces deux méthodes avec ceux décrits dans (Ammar *et al.*, 2009).

Le reste de cet article est organisé comme suit. La section 2 rappelle brièvement le principe d'apprentissage de mélanges aléatoires de modèles et la section 3 décrit les heuristiques proposées. La section 4 passe en revue différents résultats obtenus et la section 5 finit par conclure et présenter quelques perspectives.

2. Mélanges aléatoires d'arbres de Markov

Les mélanges d'arbres de Markov aléatoires ont été étudiés dans (Ammar *et al.*, 2009) et appliqués comme alternatives des méthodes classiques de l'estimation de densité dans le contexte des espaces de grandes dimensions et des ensembles de données de taille réduite.

Dans l'espace des structures d'arbres de Markov, l'inférence probabiliste (Pearl, 1986) et l'apprentissage des paramètres sont de complexité linéaire en fonction du nombre de variables n.

L'apprentissage des modèles d'arbres de Markov peut être effectué efficacement avec l'algorithme Chow et Liu qui est seulement quadratique en fonction du nombre de variables.

Soit $X = \{X_1, \ldots, X_n\}$ un ensemble fini de variables aléatoires discrètes, et $D = (x^1, \cdots, x^d)$ un ensemble de données (échantillons) d'observations jointes $x^i = (x^i_1, \cdots, x^i_n)$ générées indépendamment par une densité $\mathbb{P}_G(X)$.

Une distribution de mélange $\mathbb{P}_{\hat{\mathcal{T}}}(X_1, \ldots, X_n)$ sur un ensemble $\hat{\mathcal{T}} = \{T_1, \ldots, T_m\}$ de m arbres de Markov est définie comme une combinaison linéaire convexe de densités élémentaires d'arbres de Markov.

$$\mathbb{P}_{\hat{\mathcal{T}}}(X) = \sum_{i=1}^{m} \mu_i \mathbb{P}_{T_i}(X), \qquad [1]$$

où $\mu_i \in [0, 1]$ et $\sum_{i=1}^{m} \mu_i = 1$.

Notre procédure générique d'apprentissage d'une distribution de mélange aléatoire d'arbres de Markov à partir d'un ensemble de données D est décrite par l'algorithme 1 (Ammar *et al.*, 2009).

Cet algorithme retourne les m modèles d'arbres, ainsi que leurs paramètres θ_{T_i} et les poids des arbres μ_i.

Algorithme 1 (Apprentissage de mélange d'arbres de Markov)

1) Repeat for $i = 1, \cdots, m$ *:*

a) Draw random number ρ_i*,*

b) $T_i = \textit{DrawMarkovtree}(D, \rho_i)$*,*

c) $\tilde{\theta}_{T_i} = \textit{LearnPars}(T_i, D, \rho_i)$

2) $(\mu)_{i=1}^m = \textit{CompWeights}((T_i, \tilde{\theta}_{T_i}, \rho_i)_{i=1}^m, D)$

3) Return $\left(\mu_i, T_i, \tilde{\theta}_{T_i}\right)_{i=1}^m$.

Certaines versions des procédures de cet algorithme utilisées dans nos expérimentations sont détaillées plus précisément dans (Ammar *et al.*, 2009). Nous nous concentrons dans ce travail sur la procédure *DrawMarkovtree* et nous décrivons dans la section suivante de nouvelles heuristiques basées sur le principe du "Perturber et Combiner" en vue de réduire la complexité des approches précédemment proposées.

3. Heuristiques sub-qudratiques de la procédure *DrawMarkovtree*

Nous avons proposé dans (Ammar *et al.*, 2009) des variantes pour la procédure *DrawMarkovtree*. La première génère aléatoirement des arbres de Markov (en échantillonnant selon une loi uniforme dans l'ensemble de tous les arbres de Markov). La deuxième construit des structures d'arbres optimales en appliquant l'algorithme MWST (Maximum Weight Spanning Tree) d'apprentissage de structure sur une copie bootstrap de l'ensemble des données d'apprentissage initial. Nous avons démontré dans ce travail précédent que la meilleure méthode est celle qui utilise le bagging de structures d'arbres et qui a une complexité quadratique.

Ainsi, nous proposons d'étudier cette méthode et essayer d'accélérer la procédure d'apprentissage en conservant des performances similaires. Cette procédure peut être découpée en trois étapes : la première consiste à calculer l'information mutuelle entre chaque paire de variables pour remplir une matrice MI d'information mutuelle de taille $n \times n$, la deuxième consiste en la recherche de l'arbre de recouvrement maximal en utilisant l'algorithme MWST (nous utilisons l'algorithme de Kruskal (Cormen *et al.*, 1994)), et enfin la troisième consiste en l'apprentissage des paramètres de la structure trouvée à l'étape 2.

La première étape est quadratique en fonction du nombre de variables (n^2 termes à calculer) tandis que la deuxième étape est de complexité $E\,log(E)$ où E est le nombre d'arcs considérés. Si la matrice MI est entièrement remplie ($E = n(n-1)/2$ termes indépendants), la complexité de la seconde étape est en $n^2\ log(n)$. La troisième étape est linéaire en fonction du nombre de variables.

Afin d'obtenir une bonne estimation de densité, nous développons une approche combinant des modèles "faibles", arbres de structures aléatoires d'un côté, et arbres de structure optimale déterminés grâce à la matrice MI de l'autre. Nous nous proposons

ici d'appliquer encore une fois le principe du "Perturber et Combiner" en construisant des modèles intermédiaires appris a partir de matrices MI incomplètes. Le nombre K de termes à considérer est alors un paramètre important pour estimer la complexité totale de la procédure.

Cette procédure est décrite par l'algorithme 2.

Algorithme 2 (Procedure sous-quadratique naive DrawMarkovTree)

1) $MI_i = [\,]_{n \times n}$
2) $D_i = GenSamples(D, i)$,
3) Repeat for $k = 1, \cdots, K$ *:*
  a) Draw random pair of number (i_1, i_2),
  b) $MI_i[i_1, i_2] = ComputeMI(X_{i_1}, X_{i_2})$
4) $T_i = CompKruskal(MI_i)$,
5) Return T_i.

$\{(i_1, i_2)\}$ represente un ensemble de paires d'indices générés aléatoirement selon une distribution uniforme. Cet ensemble servira à remplir partiellement la matrice MI_i par *ComputeMI*. *CompKruskal* prend en paramètres la matrice partiellement remplie MI_i et construit l'arbre de recouvrement maximal correspondant qui sera retourné par l'algorithme.

Différentes valeurs de ce paramètre peuvent être considérées. Afin d'étudier l'intérêt de l'approche, nous nous concentrons dans ce travail sur des simulations et des résultats obtenus avec une valeur spécifique de ce paramètre $K = n\ log(n)$. Si $K = n\ log(n)$, alors la complexité de la première étape sera $n\ log(n)$. Dans la deuxième étape, nous considèrerons $E = n\ log(n)$ arcs pour calculer l'arbre de recouvrement maximal correspondant, et alors la complexité de cette étape est $E\ log(E) = n\ log(n)\ log(n\ log(n))$, inférieure au quadratique et même très proche du quasi-linéaire.

Une autre idée est considérée pour construire un arbre sous optimal de recouvrement maximal par la procédure *DrawMarkovtree*. Cette idée consiste à tirer avantage du résultat obtenu dans l'itération précédente pour calculer le prochain résultat. Les indices des arêtes de l'arbre de Markov construit à l'itération i seront utilisés en premiers pour remplir la matrice MI_{i+1} de l'itération suivante $i + 1$, ensuite nous complétons les indices des K termes en les générant aléatoirement. Cette idée peut être décrite par l'algorithme 3.

Algorithme 3 (Procédure sous-quadratique intelligente DrawMarkovTree)

1) $MI_i = [\,]_{n \times n}$
2) $D_i = GenSamples(D, i)$,

3) Repeat for $k = 1, \cdots, nbEdges(T_{i-1})$ *:*

a) $(i_1, i_2) = GetIndices(GetEdge(T_{i-1}, k)$*,*

b) $MI_i[i_1, i_2] = ComputeMI(X_{i_1}, X_{i_2})$

4) Repeat for $k = 1, \cdots, K - nbEdges(T_{i-1})$ *:*

a) Draw random pair of number (i_1, i_2)*,*

b) $MI_i[i_1, i_2] = ComputeMI(X_{i_1}, X_{i_2})$

5) $T_i = CompKruskal(MI_i)$*,*

6) Return T_i*.*

Nous considérons deux variantes de la fonction *GenSample* utilisée dans l'étape 2. des algorithmes 2 et 3. La première variante utilise l'ensemble initial des données dans chaque itération. La deuxième génère des copies bootstrap de l'ensemble initial.

Finalement, nous considérons deux variantes pour la fonction *CompWeights* proposées dans (Ammar *et al.*, 2009), nommées "Uniform weighting" (où les coefficients sont définis par $\mu_i = \frac{1}{m}, \forall i = 1, \ldots, m$) et "Bayesian averaging" (où les coefficients μ_i sont proportionnels à la probabilité a posteriori de la structure d'arbre de Markov T_i, dérivé de son score BDeu calculé sur la totalité de l'ensemble d'individus D).

4. Simulations empiriques

4.1. *Protocole*

Pour évaluer les différentes heuristiques proposées pour améliorer la complexité de la procédure de recherche, nous avons effectué des expériences pour différentes densités de probabilité cibles. Le protocole de test est le suivant.

Choix des distributions cibles Toutes nos expérimentations ont été effectuées avec des modèles ayant un ensemble de $n = 1000$ variables aléatoires binaires. Les distributions cibles $\mathbb{P}_G(X)$ utilisées sont générées à partir de réseaux bayésiens de structure complexe (graphe orienté sans circuit -DAG- quelconque). Nous utilisons les algorithmes de génération aléatoire de structure et de paramètres appropriés décrits dans (Ammar *et al.*, 2008) pour générer cette structure et ses paramètres.

Génération des ensembles de données Pour chaque densité cible et chaque taille de l'ensemble de données, nous générons 10 ensembles de données différents à l'aide de simulations de type Monte-Carlo. La taille des ensembles d'échantillons considérée dans ce travail est $N = 1000$ éléments. Etant donné le nombre total 2^n des configurations possibles de nos n variables aléatoires, nous nous intéressons aux ensembles de données de taille assez réduite.

Apprentissage des mélanges Pour un ensemble de données et une variante de l'algorithme d'apprentissage des mélanges, nous générons un ensemble de modèles de taille croissante, respectivement $m = 1$, $m = 10$, et allant jusqu'à $m = 150$ par

Nom des variantes	Génération d'arbres	Données	Coefficients	Complexité
MTU	Aléatoire	D	Uniforme	Linéaire
MTBDeu	Aléatoire	D	BDeu	Linéaire
FBU	Algo 2	B	Uniforme	Sous-quadratique
FBBDeu	Algo 2	B	BDeu	Sous-quadratique
FDU	Algo 2	D	Uniforme	Sous-quadratique
FDBDeu	Algo 2	D	Uniforme	Sous-quadratique
FRBU	Algo 3	B	Uniforme	Sous-quadratique
FRBBDeu	Algo 3	B	BDeu	Sous-quadratique
FRDU	Algo 3	D	Uniforme	Sous-quadratique
FRDBDeu	Algo 3	D	BDeu	Sous-quadratique
CL	MWST	D	Uniforme	quadratique

Tableau 1. *Les variantes de l'algorithme générique*

incrément de 10. Cela nous permet d'observer l'effet de la taille du mélange sur la qualité du modèle résultant.

Evaluation de la qualité de l'estimation La qualité d'une densité induite par un algorithme d'apprentissage sur un ensemble de données est évaluée par la divergence KL approchée de Kullback-Leibler (Kullback *et al.*, 1951) entre cette densité estimée et la densité cible $\mathbb{P}_G(X)$ avec laquelle les données ont été générées.

Implémentation Nos différents algorithmes ont été implémentés en C++ avec la librairie Boost (`http ://www.boost.org/`) et les API fournis par la plateforme ProBT© (`http ://bayesian-programming.org`).

Notre algorithme générique peut se décliner en faisant varier la fonction de génération des arbres (aléatoirement, algorithmes 2 ou 3), l'ensemble de données d'apprentissage (les données initiales D ou une copie bootstrap B) et les coefficients du mélange (uniformes ou BDeu).

Le tableau 1 résume le nom des différentes variantes que nous comparons dans la section suivante.

4.2. *Résultats*

La Figure 1 présente un ensemble de courbes d'apprentissage correspondant à nos simulations. L'axe horisontal correspond au nombre m de termes du mélange, tandis que l'axe vertical correspond aux valeurs de KL par rapport à la densité cible. Toutes les courbes représentent des moyennes des résultats obtenus pour 10 ensembles de données d'apprentissage différents de taille 1000 et pour sept distributions cibles.

Nous observons a partir de la Figure 1 que nos deux méthodes de mélanges d'arbres sous-optimales sont nettement meilleures que les méthodes de mélanges

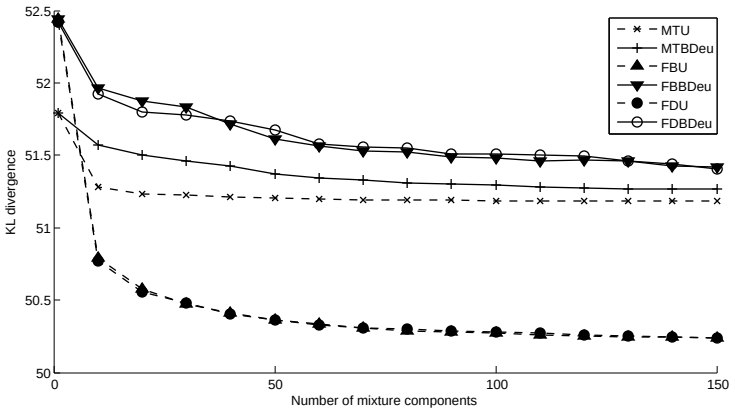

Figure 1. *Mélanges sous-quadratiques naifs d'arbres pour l'estimation de la densité d'une distribution cible de type DAG. 10 itérations avec 1000 échantillons pour 7 distributions cibles aléatoires de 1000 variables.*

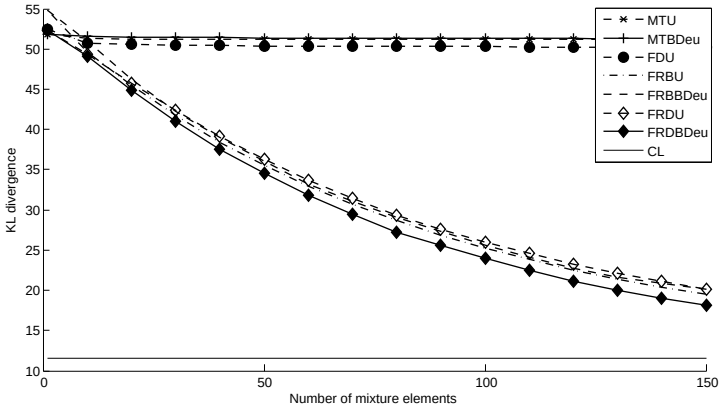

Figure 2. *Mélanges sous-quadratiques intelligents d'arbres pour l'estimation de la densité d'une distribution cible de type DAG. 10 itérations avec 1000 échantillons pour 7 distributions cibles aléatoires de 1000 variables.*

d'arbres de Markov aléatoires MTU et $MTBDeu$ du point de vue performance lorsque nous utilisons un schéma de coefficients uniformes (FDU et FBU), mais légèrement moins bon dans le cas de coefficients non uniformes.

En ce qui concerne notre seconde proposition, nous observons à partir de la Figure 2 que toutes les variantes sont nettement meilleures en terme de qualité d'estimation que les méthodes aléatoires de mélanges d'arbres de Markov (qui sont linéaires

en fonction du nombre de variables) et approchent la méthode de base CL (qui est quadratique en fonction du nombre de variables) lorsque le nombre de composantes dans le mélange augmente. Nous notons aussi qu'avec toutes ces méthodes sous-quadratiques le schéma de poids uniforme est meilleur que celui utilisant des poids non uniformes basés sur la probabilité a posteriori des structure sachant les données.

Enfin, nous notons que le principe du Bagging ne permet pas d'obtenir des résultats meilleurs qu'en utilisant l'ensemble de données original dans ce contexte de grandes dimensions et peu de données. En conclusion, la meilleure méthode dans ces simulations est l'approche "intelligente" qui utilise un mélange uniforme d'arbres sous-optimaaux construits en utilisant l'algorithme de Chow et Liu sur une matrice d'information mutuelle partiellement remplie, dont les termes ne sont pas générés totalement à l'aléatoire, et l'ensemble original de données.

De point de vue complexité, les méthodes que nous proposons sont de complexité est $n\ log(n) \log(n \log(n)$ (sous-quadratique) et permettent d'obtenir de meilleurs résultats que les méthodes linéaires tout en approchant la méthode de base CL qui est quadratique.

5. Conclusions et perspectives

Nous avons proposé dans cette étude d'appliquer le principe du "Perturber et Combiner" à l'estimation de la densité dans le contexte de l'apprentissage non supervisé avec les modèles graphiques probabilistes. Nous utilisons des modèles sous-optimaux construits à partir d'une matrice MI incomplète et de l'algorithme de Chow et Liu. Nous avons ainsi présenté deux heuristiques de complexité algorithmique sous-quadratique. La perturbation est présente à plusieurs niveaux, tout d'abord avec le remplissage partiel de la matrice MI en générant aléatoirement une partie de ses termes, mais aussi avec la génération d'une copie Bootstrap des données servant au calcul d'information mutuelle.

Le résultat le plus intéressant est que notre seconde proposition, l'approche "intelligente" de complexité $n\ log(n) \log(n \log(n))$, très proche du quasi-linéaire, permet d'avoir des résultats nettement meilleurs que ceux obtenus par les mélanges aléatoires d'arbres de Markov qui sont linéaires et approche la méthode CL qui est quadratique en fonction du nombre de variables.

Nos perspectives pour ce travail sont d'améliorer la complexité de nos méthodes en utilisant des approximations linéaires de l'arbre de poids maximal (Chazelle, 2000). Nous pensons aussi que la combinaison de nos approches avec des méthodes séquentielles, telles que le Boosting ou Markov-Chain Monte-Carlo qui sont déjà appliqués dans le contexte des modèles graphiques probabilistes, pourrait constituer une avenue très riche pour la construction de nouveaux algorithmes d'estimation de densité.

6. Bibliographie

Ammar S., Leray P., Defoumy B., Wehenkel L., « High-Dimensional Probability Density Estimation with Randomized Ensembles of Tree Structured Bayesian Networks », *Proceedings of the fourth European Workshop on Probabilistic Graphical Models (PGM'08)*, Hirtshals, Denmark, p. 9-16, 2008.

Ammar S., Leray P., Defoumy B., Wehenkel L., « Probability Density Estimation by Perturbing and Combining Tree Structured Markov Networks », *Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2009)*, Verona, Italy, p. 156-167, 2009.

Chazelle B., « A minimum spanning tree algorithm with inverse-Ackermann type complexity », *J. ACM*, vol. 47, n° 6, p. 1028-1047, 2000.

Chow C., Liu C. N., « Approximating discrete probability distributions with dependence trees », *IEEE Transactions on Information Theory*, vol. 14, n° 3, p. 462-467, 1968.

Cormen T., Leiserson C., Rivest R., *Introduction à l'algorithmique*, Dunod, 1994.

Cowell R., Dawid A., Lauritzen S., Spiegelhalter D., *Probabilistic Networks and Expert Systems*, Statistics for Engineering and Information Science, Springer-Verlag, 1999.

Geurts P., Ernst D., Wehenkel L., « Extremely Randomized Trees », *Machine Learning*, vol. 63, n° 1, p. 3-42, 2006.

Kullback S., Leibler R., « On Information and Sufficiency », *Annals of Mathematical Statistics*, vol. 22, n° 1, p. 79-86, 1951.

Pearl J., « Fusion, Propagation, and Structuring in Belief Networks », *Artificial Intelligence*, vol. 29, p. 241-288, 1986.

Ridgeway G., « Looking for lumps : boosting and bagging for density estimation », *Computational Statistics & Data Analysis*, vol. 38, n° 4, p. 379 - 392, 2002.