Efficiently Approximating Markov Tree Bagging for High-Dimensional Density Estimation

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

[1] Université de Liège, Department of EECS and GIGA-Research, Grande Traverse, 10 - B-4000 Liège - Belgium
{fschnitzler,P.Geurts,L.Wehenkel}@ulg.ac.be
[2] Ecole Polytechnique de l'Université de Nantes, Knowledge and Decision Team, Laboratoire d'Informatique de Nantes Atlantique (LINA) UMR 6241, France
{sourour.ammar,philippe.leray}@univ-nantes.fr

Abstract. We consider algorithms for generating *Mixtures of Bagged Markov Trees*, for density estimation. In problems defined over many variables and when few observations are available, those mixtures generally outperform a single Markov tree maximizing the data likelihood, but are far more expensive to compute. In this paper, we describe new algorithms for approximating such models, with the aim of *speeding up learning without sacrificing accuracy*. More specifically, we propose to use a filtering step obtained as a by-product from computing a first Markov tree, so as to avoid considering poor candidate edges in the subsequently generated trees. We compare these algorithms (on synthetic data sets) to Mixtures of Bagged Markov Trees, as well as to a single Markov tree derived by the classical Chow-Liu algorithm and to a recently proposed randomized scheme used for building tree mixtures.

Keywords: mixture models, Markov trees, bagging, randomization.

1 Introduction

Estimation of multivariate probability densities from observational data is a widely used strategy to tackle decision making problems under uncertainty. A density model can be used to answer various queries about the underlying data generation mechanism (also called performing inference), such as computing the likelihood of observing a problem instance, or estimating the conditional probability density of a subset of variables given observed values of another subset.

The framework of probabilistic graphical models [18,28] provides well founded approaches to model probability densities and to perform inference by combining graph theory, with statistics and algorithmics. The structure of a graphical model encodes relationships between variables while its parameters quantify those interactions. Bayesian networks are a class of models that encode a joint probability density over a set of variables by a product of conditional probability densities (see Sect. 2). Both learning and inference are however NP-hard with those models when the underlying graph is unconstrained [11,22].

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 113–128, 2011.

To cope with the problem size expansion faced today in many applications due to the rapid increase in measurement resolution, many learning methods for bayesian networks incorporate some constraints on their graphical structure, e.g. [7,14,16,31]. In that regard, an interesting subset of those models is the class of Markov trees, where each conditional probability distribution is conditioned on a single variable (except the root) [28]: learning a Markov tree maximizing the data likelihood by the Chow-Liu algorithm [10] has a computational complexity essentially quadratic in the number of variables, while performing inference with such models is of linear complexity. Another advantage of Markov trees is their small number of parameters, which reduces the risk of overfitting when data is scarce. However, for problems with very large numbers of variables and low sample size this model class may already be too large, and it may be desirable to impose additional regularization constraints on top of this method [23].

Bootstrap aggregation (bagging) [8,12] is a meta-algorithm that compensates for a lack of data by applying a given algorithm on several bootstrap replicas of the original data set and averaging the predictions of the resulting models. A bootstrap replica is obtained by randomly drawing (with replacement) original samples and copying them into the replica. Averaging the predictions from an ensemble of models derived from an ensemble of bootstrap replicas leads to a decrease in variance and hence a reduction in overfitting. This meta-algorithm, originally developed in the context of supervised learning, has already been applied for learning probabilistic graphical models e.g. [13,15], often to get a more robust structure but without consideration for inference on the said structure.

When one is willing to use bootstrapping to obtain a density model on which inference is tractable, one interesting possibility is to use bagged mixtures of Chow-Liu trees. Indeed, these have been shown to outperform single Chow-Liu trees, specially on high-dimensional problems with small sample sizes [2]. However, the extra computational cost with respect to learning one single Chow-Liu tree may prove problematic on very large problems. In this work we therefore investigate means to reduce this complexity by approximating the original bootstrap procedure. We propose to couple a first application of the Chow-Liu algorithm to either subsampling the set of candidate edges, or to a statistical test to detect irrelevant edges, in order to avoid considering all candidate edges in the subsequent runs of the Chow-Liu algorithm on subsequent bootstrap replicas. The second approach can be seen as applying a structural regularization to identify a skeleton comprising only potentially relevant edges, and then restricting the search of optimal Markov trees on subsequent bootstrap replicas within that smaller envelope instead of the complete set of all possible edges; it may hence also be beneficial in terms of accuracy in very small sample size conditions.

In Sect. 2 we cover the concept of bayesian networks, bayesian learning and mixtures of Markov trees in more details. We then describe the baseline algorithms for Markov tree based density models upon which we propose improvements (Sect. 3), before detailing our new algorithms (Sect. 4). The experiments performed to compare them in terms of accuracy, convergence speed and computing times are presented and discussed in Sect. 5.

2 Graphical Probability Density Model Learning

A bayesian network [28] is a probabilistic graphical model that encodes a joint probability density over a finite set $\mathcal{X}$ of n variables $\{X_1, X_2, ..., X_n\}$. Those variables correspond to the nodes of a Directed Acyclic Graph (DAG) $\mathcal{G}$ that encodes conditional independence relationships between variables and allows their algorithmic verification. The graphical structure actually defines a factorization of the joint density as a product of conditional densities of each variable X_i conditionally to the set of its parents $Pa_{\mathcal{G}}(X_i)$ in the graph:

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

In the case of discrete variables, learning the parameters defining those conditional densities from data is relatively straightforward, but structure learning is not. There are three main structure learning approaches: a score-based, a constraint-based and a bayesian one [18].

In the *score-based approach* [5], a numerical criterion (maximum likelihood, BIC, AIC...) is defined over the set of DAGs, and learning can be defined as selecting, among all DAGs, the one that maximizes this score with respect to the data set. However, the number of possible DAGs (as well as the number of their equivalence classes) grows superexponentially with the number of variables n [29]. Since existing unconstrained score optimization algorithms are not scalable, simplifications must in practice be used. These may be achieved by reducing the number of candidate structures, either by restricting the resolution of the search space [6] or by limiting its range (e.g. by constraining the number of candidate parents or the global structures searched [14]).

The *constraint-based approach* [1] consists in extracting from the observational data a set of conditional independence relationships (statements $\mathcal{S}_i \perp \mathcal{S}_j | \mathcal{S}_k$, where $\mathcal{S}_i, \mathcal{S}_j, \mathcal{S}_k$ are disjoint subsets of variables), and searching for a structure that best matches those constraints. Algorithms typically consider the assessment of a polynomial number of independence relationships, and achieve this by limiting the cardinalities of the subsets of variables inspected.

The *bayesian averaging approach* [24] considers the set of all possible structures rather than identifying a single best one, and averages predictions from those structures in accordance with the goal of the learning procedure. Taking into account all possible structures is rarely possible, and approximations must thus be employed. One of these strategies is the bootstrap aggregation approach that we are considering in this article (see Sect. 2.1).

But the cost of inference must also be considered. Its complexity grows with the tree-width of the DAG [22], which is the minimum size, minus one, of the largest connected subgraph in a moralized and triangularized version of the DAG (obtained by first joining all non-adjacent parents of any variable, and then by chordalizing all cycles). Although many heuristic approaches to inference have been developed, many learning methods target low tree-width structures [7,14,16,31] and thus also limit inference complexity.

Markov trees, a subclass of bayesian networks, allow for scalable learning and inference; it consists of all bayesian networks where each variable has a single parent. The Chow-Liu algorithm (Sect. 3.1) produces a Markov tree maximizing the likelihood of a data set, and its complexity is essentially quadratic in the number of variables. The tree-width of a Markov tree is always one, and inference is thus linear in the number of variables. Both properties make Markov trees extremely interesting for high-dimensional density modeling.

2.1 Bagging in the Context of Learning Bayesian Networks

Bagging is a model averaging method where a given learning algorithm is randomized by applying it on m different bootstrap replica data sets, therefore resulting in m different models. A bootstrap replica $\mathbf{D}'$ of size p' is obtained from an original data set $\mathbf{D}$ of p observations by uniformly and independently drawing p' natural numbers $r_i \in [1, p]$, and by compiling $\mathbf{D}'$ by

$$\mathbf{D}'[i] = \mathbf{D}[r_i] \qquad \forall i \in [1, p'] \ , \tag{2}$$

where $\mathbf{D}[j]$ (resp. $\mathbf{D}'[k]$) refers to jth (resp. kth) observation of $\mathbf{D}$ (resp. $\mathbf{D}'$), and where typically (as in this paper) $p = p'$.

The result of the bagging algorithm is an average between the m models learnt from the m bootstrap replicas that typically exhibits a lower variance than a model learned directly from the original data set. This approach has been quite popular and effective in the context of supervised learning [8].

Bagging has been proposed for Gaussian density modeling [27], and for structure learning of graphical models, e.g. by considering the frequency of occurrence of interesting graphical features among the structures derived from bootstrap replicas [15], and also to improve score-based structure learning by incorporating the bootstrap procedure in the computation of the score [13]. Recently, it was proposed for generating ensembles of bagged Chow-Liu trees [2]; this latter approach is denoted in the rest of this paper by *Bagged Mixture of Chow-Liu Trees*; it is discussed more in detail in the next section.

2.2 Mixtures of Markov Trees

A mixture of Markov trees over a set of n variables $\mathcal{X}$ is a convex combination of a set $\hat{\mathcal{T}} = \{T_1, \ldots, T_m\}$ of m elementary Markov tree densities, i.e.

$$P_{\hat{\mathcal{T}}}(\mathcal{X}) = \sum_{i=1}^{m} \mu_i P_{T_i}(\mathcal{X}) \ , \tag{3}$$

where $\{\mu_i\}_{i=1}^m$ are the weights of the mixture ($\mu_i \in [0, 1]$ and $\sum_{i=1}^m \mu_i = 1$). The complexity of inference in this model is thus equal to m times the complexity of inference with a single tree, which is linear in the number n of variables.

Several learning algorithms of mixtures of Markov trees have already been proposed; they can be categorized into two groups: the maximum likelihood and the randomization approaches.

In the former approach, the mixture of trees is primarily used as a mean to exploit the good algorithmic properties of trees while improving their modeling capabilities. These methods include using the EM algorithm to partition the data between a given number of terms [25], or using clever reweighting schemes on the whole data set to fit modes of the density [21].

The second approach can be viewed as an attempt to approximate true bayesian learning in the space of Markov tree structures. In these methods, a set of tree models are generated using a more or less strongly randomized procedure, that can range from completely random structures based on Prüfer lists to bagged Mixtures of trees. The weights associated to these trees can be either uniform or proportional to the score of the structure based on the data set. A comparison of theses approaches can be found in [3]. The present work adopts this strategy and some of those methods are further described in Sect. 3.

An approach at the intersection of those two categories has been proposed in [17], where a MCMC exploration scheme is defined on the space of mixtures of trees using a Dirichlet process and a suitable prior on tree structures [26].

3 Baseline Markov Tree Based Learning Algorithms

In this section, we describe the three baseline methods of density estimation with Markov trees reused in this paper, and we state their computational complexity.

3.1 The Chow-Liu Algorithm for Learning a Markov Tree

The algorithm for learning a Markov tree structure $T_{CL}(\mathbf{D})$ maximizing the likelihood of a training set $\mathbf{D}$ was introduced by Chow and Liu [10]. It solves the optimization problem

$$T_{CL}(\mathbf{D}) = \arg\max_{T} \sum_{(X_i,X_j)\in\mathcal{E}(T)} I_{\mathbf{D}}(X_i;X_j) \ , \tag{4}$$

where $\mathcal{E}(T)$ is the set of edges in T, constrained to be a tree, and where $I_{\mathbf{D}}(X_i,X_j)$ is the maximum likelihood estimate of the mutual information among variables X_i and X_j computed from the dataset $\mathbf{D}$ (composed of p observations).

Algorithm 1 has two steps: first $I_{\mathbf{D}}(X_i,X_j)\,(\forall i=1\dots n, \forall j=i+1\dots n)$ are computed to fill an $n\times n$ symmetrical matrix (MI), then used to compute a maximum weight spanning tree (MWST, e.g. by [19] as here, or by [9]).

Algorithm 1 (Chow-Liu (CL) tree)

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}*.*

Step 2 requires $\mathcal{O}(n^2p)$ computations, while computing a MWST has a complexity of $E\log(E)$ with E the number of candidate edges. Here $E = n(n-1)/2$, so that, for fixed sample size p, the complexity is $\mathcal{O}(n^2\log(n^2)) \equiv \mathcal{O}(n^2\log(n))$.

3.2 Bagging of Chow-Liu Markov Trees

Bagging of the Chow-Liu algorithm is described by Algorithm 2.

Algorithm 2 (Generating a mixture of bagged Chow-Liu trees)

1. $\hat{\mathcal{T}} = \{\}$
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* $\hat{\mathcal{T}}$, $\mu = \{1/m, \cdots, 1/m\}$.

The complexity of Algorithm 2 is m times the complexity of the Chow-Liu algorithm, or $\mathcal{O}(mn^2 \log(n))$, for fixed sample size p.

Notice that it was shown in [30] that learning the parameters of each tree in $\hat{\mathcal{T}}$ on $\mathbf{D}$ rather than on the replica $\mathbf{D}'$ used to generate its structure improves accuracy, and we will therefore use $\mathbf{D}$ to estimate the parameters of all Markov trees generated by all the algorithms studied in this paper, according to [30].

3.3 Inertial Search Heuristic

This algorithm [4] improves the computational complexity of the Bagging method by limiting to a specified number K the number of variable pairs and mutual informations computed and considered for each MWST construction. Constructing $\hat{\mathcal{T}}$ is done here by a sequential procedure: for optimizing the first tree, a random subset of K edges is considered, and then for each subsequent tree T_i, the considered subset is initialized by the edges of the previous tree, $\mathcal{S} = \mathcal{E}(T_{i-1})$, and completed with an additional random subset of $K - |\mathcal{E}(T_{i-1})|$ edges.

Algorithm 3 (Inertial search of mixtures of Markov trees (ISH))

1. $\hat{\mathcal{T}} = \{\}$, $\mathcal{S} = \{\}$
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* $\hat{\mathcal{T}}$, $\mu = \{1/m, \cdots, 1/m\}$.

The parameter K controls the computational complexity of the method, which is $\mathcal{O}(mK \log K)$. We use $K = Cn \ln n$ as in [4] (with $C = 1$ in most of our simulations) leading to a complexity approximately of $\mathcal{O}(mn \log(n))$.

4 Proposed Algorithms

In this section we propose our alternative algorithms. They all start by computing a Chow-Liu tree on the original data set $\mathbf{D}$ and they then use the results of this computation for accelerating the generation of subsequent ensemble terms.

4.1 Improving the Inertial Search Heuristic by Warm Start

While Algorithm 3 is of log-linear complexity in n and gradually improves as new trees are added to the model, it consists essentially in an exploration of the matrix MI of mutual informations. Notice that without bagging (i.e. by using $\mathbf{D}' = \mathbf{D}$ at all iterations), this algorithm would eventually converge to the Chow-Liu tree, since Tarjan's red rule [32] implies that the lightest edge of any cycle is not part of the MWST. However, the number of iterations needed to fully explore the matrix essentially increases with n, since

$$\lim_{n\to\infty} \frac{\text{Edges considered at each iteration}}{\text{Total edges}} = \frac{\mathcal{O}(n \log n)}{\mathcal{O}(n^2)} = 0 \ , \tag{5}$$

and hence the algorithm will take longer and longer to converge as n increases (see also our experimental results in the next section).

We hence modified this method, by changing the first iteration so as to start with a more optimal set of edges ($\mathcal{E}(T_{CL})$ computed by the Chow-Liu algorithm based on a complete matrix of mutual informations; see Algorithm 4).

Algorithm 4 (Warm start inertial research procedure (Warm start ISH))

1. $\hat{T} = \{\textbf{Chow-Liu}(\mathbf{D})\}$, $\mathcal{S} = \mathcal{E}(\textbf{Chow-Liu}(\mathbf{D}))$
2. *Repeat for* $j = 2, \cdots, m$*:*
   $\cdots$ *(identical to points (a) to (g) of Algorithm 3)*
3. *Return* $\hat{T}$, $\mu = \{1/m, \cdots, 1/m\}$.

The complexity of this method is $\mathcal{O}(n^2 \log(n) + mK \log(K))$ where K is the number of edges considered at each iteration after the first one. As in Algorithm 3, we set $K = Cn \ln n$. In practice the gain in convergence speed strongly compensates for the increased complexity needed for computing the first term (see our results in the next section).

Alternatively, both methods could be viewed as a stochastic walk in the space of Markov tree structures that at convergence will attain the set of good structures. Algorithm 3 however starts very far from this set while the variant we propose in Algorithm 4 starts from a more sensible initial guess.

4.2 Pruned Mixtures of Bagged Chow-Liu Trees

The Chow-Liu method (Algorithm 1), and its bagging (Algorithm 2) compute connected Markov tree structures of maximum likelihood over the data sets they get as input. But in high-dimensional problems (with $p \ll n$), maximizing the

data likelihood over all possible tree structures may already lead by itself to overfitting. We therefore consider a *structural* regularization of the Chow-Liu method, by modifying its optimization criterion of eqn. (4), so as to penalize model complexity in terms of its number of edges $|T|$,

$$T_{CL}^{\lambda}(\mathbf{D}) = \arg\max_{T} \sum_{(X_i,X_j)\in\mathcal{E}(T)} I_{\mathbf{D}}(X_i;X_j) - \lambda|T| \ , \tag{6}$$

where T is now allowed to be a forest (*at most* one path between any two nodes).

The optimal solution to this problem can be obtained by modifying the greedy Chow-Liu algorithm, to return the 'forest model' as soon as the next edge to be included provides an information quantity $I_{\mathbf{D}}(X_i;X_j)$ smaller than λ. Furthermore, as for supervised decision tree growing [33], we notice that penalizing in this way the tree complexity is tantamount to using a hypothesis test for checking independence of the next pair of variables to be included; such a test may be formulated by comparing the quantity $2p(\ln 2)I_{\mathbf{D}}(X_i;X_j)$ (χ-square distributed under independence, with a degree of freedom of 1 for binary variables) to a critical value depending on a postulated p-value, say $\alpha = 0.05$ or smaller. This means that an arc relating to a pair of variables (X_i, X_j) such that $2p(\ln 2)I_{\mathbf{D}}(X_i;X_j)$ computed from the dataset is smaller than the χ-square statistic threshold computed for α will never be included in the forest by our modified algorithm.

To take advantage of the first iteration of the algorithm, we use the computations performed for building a first tree of the mixture by the Chow-Liu algorithm to identify those pairs of variables whose mutual information is above the threshold, and we then consider only the set $\mathcal{S}$ of those latter pairs of variables for building trees composing the rest of the mixture (see Algorithm 5).

Algorithm 5 (Pre-pruned (bagged) Chow-Liu trees (PMBCL))

1. $\mathcal{S} = \{\}$, $MI = [0]_{n\times n}$
2. *Repeat for* $i_1, i_2 > i_1$, $i_1, i_2 \in 1, \cdots, n$
   if $I_{\mathbf{D}}(X_i;X_j) > \lambda(\alpha)$
   (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* $\hat{\mathcal{T}}$, $\mu = \{1/m, \cdots, 1/m\}$.

The complexity of Algorithm 5 is $\mathcal{O}(n^2 + mK(\alpha)\log(K(\alpha))$, i.e. similar to that of Algorithm 4 (where $K = n\ln n$); its first term is also independent of the mixture size m and its second term now depends on the effect of the chosen value of α on the number of candidate edges $K(\alpha) \equiv |\mathcal{S}|$ retained in the skeleton (the smaller α, the smaller $K(\alpha)$, in a dataset size dependent fashion).

5 Experiments

Here we empirically compare our algorithms of Sect. 4 to the baseline methods of Sect. 3. To this end, we use simulated target densities that are represented by synthetic bayesian networks over binary variables. Each structure is randomly drawn by considering variables sequentially, by uniformly drawing the number of parents for each X_i in $[0, max(5, i-1)]$ and by randomly selecting these parents in $\{X_1, ..., X_{i-1}\}$. Parameters of the networks are drawn from uniform Dirichlet distributions [30]. We present results with $n = 200$ or $n = 1000$ variables, and we performed our analysis based on data sets of $p = 200, 600, 1000$ observations, i.e. small samples given the number of variables. All results are averaged over 5 target densities and 6 learning sets for each density.

We focussed the analysis on the merit of the estimation of the probability distribution. We assessed the quality of each generated mixture by the Kullback-Leibler divergence [20], an asymmetric measure of similarity of a given density $P_{\hat{\mathcal{T}}}$ to a target density P, defined by

$$D_{KL}(P \mid\mid P_{\hat{\mathcal{T}}}) = \sum_{X \in \mathcal{X}} P(X) \log_2 \left(\frac{P(X)}{P_{\hat{\mathcal{T}}}(X)} \right) \quad . \tag{7}$$

But for computational reasons (considering all 2^n possible configurations of $\mathcal{X}$ is not feasible) this score was approximated by a Monte-Carlo procedure:

$$\hat{D}_{KL}(P \mid\mid P_{\hat{\mathcal{T}}}) = \frac{1}{N} \sum_{X \sim P}^{N} \log_2 \left(\frac{P(X)}{P_{\hat{\mathcal{T}}}(X)} \right) \quad , \tag{8}$$

where we used one test set of 50000 independent observations to estimate all the models inferred for a given target density.

For a given data set, we applied all algorithms and compared their results to the target density. Except for the Chow-Liu algorithm that produces a single tree, all mixture models are evaluated for growing numbers of terms (m=1,10,20...) up to 500 for 200 variables, in order to assess the convergence of the different methods (especially Algorithm 3), and up to 200 trees for 1000 variables to investigate the impact of the number of variables.

Parameters of all trees are learned from the full training sets (i.e. not from the bootstrap replicas that are used only to generate the structures), by maximizing the posterior likelihood of the data set based on uniform Dirichlet priors [30].

5.1 Results in Terms of Accuracies

Let us start by an evaluation of the relative accuracy performances of the different algorithms in the case of 1000 variables and 200 observations. Figure 1 displays the Kullback-Leibler divergence (vertical axis) with respect to the target density for the single CL tree (Algorithm 1) and for the other methods as a function of the mixture size m (horizontal axis). The PMBCL method (Algorithm 5) is tested here with 4 values of its parameter α: $1E^{-1}, 5E^{-2}, 5E^{-3}, 5E^{-4}$.

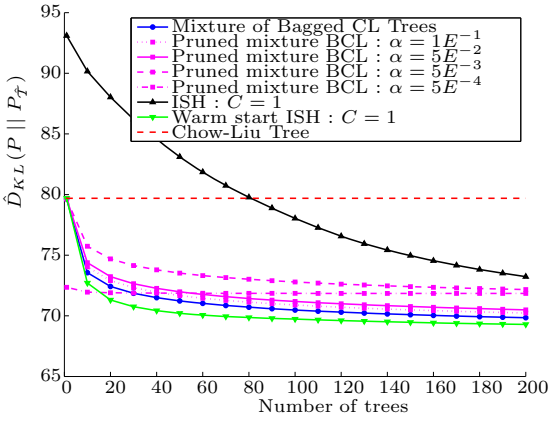


Fig. 1. A comparison between all methods presented in this paper shows the superiority of model averaging methods (with $n = 1000$ and $p = 200$). Horizontal axis: ensemble size m; vertical axis KL divergence to the target density estimated by Monte-Carlo and averaged over 5 target densities and 6 training sets.

Looking first at $m = 1$ (initial values of all curves), we observe that all but two methods start at the same point as the CL tree: ISH (Algorithm 3) is significantly worse, while the strongly regularized PMBCL tree at $\alpha = 5E^{-4}$ is significantly better[1]. For larger values of m, all the considered mixtures monotonically improve, some more quickly than others, and for sufficiently high values of m they all are quite superior to a single CL tree. For PMBCL, the smaller α, the lesser the improvement rate; actually, for $\alpha = 5E^{-4}$, its improvement rate is so small that it is quickly overtaken by the Mixture of Bagged CL Trees (at $m = 30$) and later on by PMBCL with $\alpha = 5E^{-2}$ (at $m = 60$). On the other hand, Warm Start ISH and PMBCL for α sufficiently large display comparable performances and the same convergence rate than Bagged CL Trees.

These results confirm the superiority of the model averaging approach, and they also suggest the interest of trying to limit the complexities of the individual trees in PMBCL and to correctly initialize the inertial approach, given their computational complexity advantage with respect to raw bagging (see below).

In order to allow a better understanding of the influence of α on the behavior of PMBCL, Table 1 lists the number of edges in the skeleton S and in the first tree T_1 for different values of α. It comes as no surprise that those numbers are decreasing with α. Note how the number of edges in T_1 is almost at the maximum ($n - 1 = 999$) for $\alpha \in \{1E^{-1}, 5E^{-2}, 5E^{-3}\}$, whose curves start at the same performance as the CL tree, while the smallest α ($5E^{-4}$) leads to a much smaller tree. These numbers also show that the skeleton in that last case has only a few edges more than the first tree. This is in accordance with the very small improvement in the performance of the method when the mixture

[1] Standard deviations of KL divergences, not reported for the sake of legibility, are about 20 times smaller than the average differences that we comment.

Table 1. Impact of the parameter α on the number of edges in PMBCL, averaged on 5 densities times 6 data sets for $n = 1000$ variables and $p = 200$ samples

	Numbers (% of the total) for $\alpha =$			
	$1E^{-1}$	$5E^{-2}$	$5E^{-3}$	$5E^{-4}$
Edges in T_1	998	997.9	993.2	626.8
Edges in S	52278(10.5%)	26821(5.36%)	3311(0.66%)	683 (0.13%)

is expanded and its fast convergence: the skeleton is so small that only a few different trees can be built with those edges, and the mixture quickly has them.

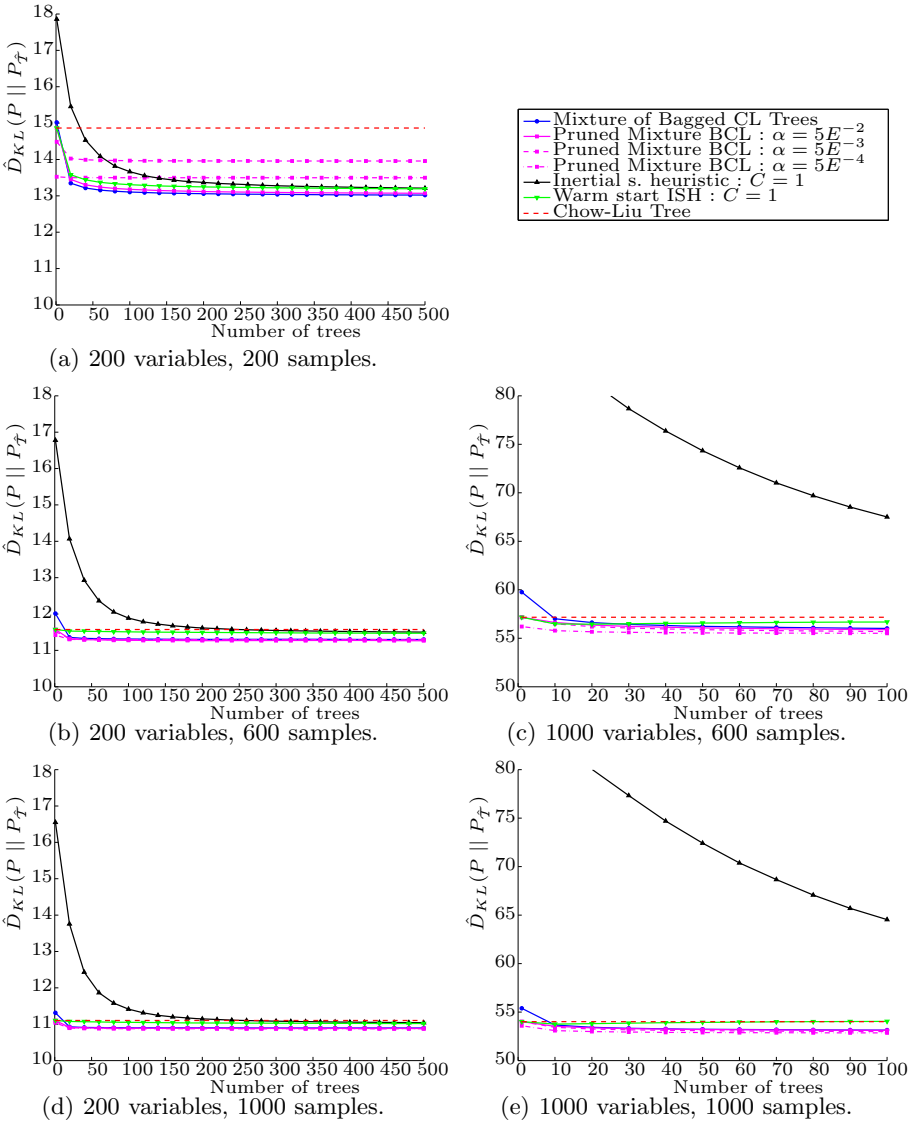
On the other hand, when the skeleton is larger, tree structures learned on bagged replica have more freedom, which allows the consideration of more candidate structures and leads to a more effective variance reduction.

Effect of the Learning Set Size *p*. To further analyze the relative behavior of the different methods, we increased the size p of learning samples to 600 and 1000. Results, reported in Figs 2(c,e), show that the most noticeable change is that the different methods now start from different initial points: the CL tree becomes initially better than a tree learned on a bagged replica. The advantage of the first PMBCL tree with the smallest value α is decreasing. We deem that both observations are a consequence of the improved precision of the mutual information estimate derived from a larger data set.

Now that the estimations of the "good" edges are better, reducing α seems to have an opposite effect on the improvement rate of PMBCL. Notice that, while at $m = 100$, the lowest α still seems better on average, confidence intervals (not displayed) suggest that the different methods cannot really be distinguished.

The Warm Start ISH is now doing far worse than the Mixture of Bagged CL Trees. We conjecture that the larger sample size leads to less variation in the mutual informations computed from bootstrap replicas, leading to slower moves in the space of tree structures for this method.

Effect of the Problem Dimensionality *n*. Modifying the number of variables has mostly an effect on the ISH methods, since it impacts the relative number of edges considered at each iteration, and thus the exploration speed of the MI matrix. Smaller numbers of variables therefore should accelerate the convergence of this method. Figures 2(a,b,d), provide a global picture of the relative performances of the considered methods, with $n = 200$ and over a longer horizon $m = 500$ of averaging. These simulations show that both inertial methods converge to the same point. Therefore, and despite a better improvement rate at the beginning, sampling structures far from the optimal one (in the original ISH method) does not improve the mixture. Based on this observation, one might actually be tempted to remove the first terms of that mixture, hoping for an improved convergence speed. We however believe that considering all edges in the first step of the method (the Warm Start variant) is more productive, since the method is directly initialized in the neighborhood of good structures.



(a) 200 variables, 200 samples.

(b) 200 variables, 600 samples.

(c) 1000 variables, 600 samples.

(d) 200 variables, 1000 samples.

(e) 1000 variables, 1000 samples.

Fig. 2. Overview of accuracy performances of the different algorithms described in this paper, with $n = 200$ or $n = 1000$ variables (left vs right), and for increasing sample sizes p (200, 600 and 1000, from top to bottom). Vertical axis: KL divergence to the target density estimated by Monte Carlo on 50,000 test observations, averaged over 5 target densities and 6 learning sets for each one. Horizontal axis: number m of mixture terms used by the different methods (except for the CL tree method, using a single tree).

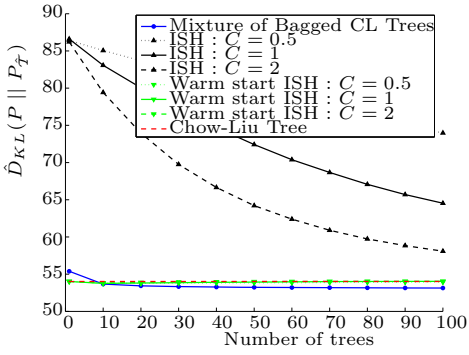


Fig. 3. Modifying the number of edges considered at each step only affects ISH when all edges are not considered in the first iteration (shown here for $n = 1000$, $p = 1000$)

Inertial Search Heuristics. Modifying the number of edges considered at each iteration in both variants of ISH, as depicted in Fig. 3, shows that the exploration of the MI matrix affects the convergence of the base method. Indeed, doubling ($C = 2$) or dividing by two ($C = 0.5$) the number of edges explored has a huge impact on its convergence, while it hardly affects the Warm Start version.

5.2 Computing Times

Our experiments were performed on a grid running ClusterVisionOS and composed of pairs of Intel L5420 2.50 Ghz processors with either 16 or 32 GB of RAM. Due to the environment, run time for a method can vary a lot, and we therefore decided to report relative minimum running time for every method. Those results are displayed in Table 2 and 3 for respectively 200 and 1000 variables / 500 and 100 trees. Results for PMBCL are reported for $\alpha = 0.005$.

Those numbers show that the proposed methods (lower part of the table) are roughly an order of magnitude faster than the standard bagging method, and this relative speed-up is stronger in the higher dimensional case. Also, as we saw from the accuracy results, these methods converge as quickly as bagging.

If one is considering parallelizing those methods at a high level, namely by computing trees individually on different cores, Bagged mixtures of CL Trees and PMBCL are the best candidates, since the trees in these methods are independent (independent conditionally on the first tree in the case of PMBCL). In the two ISH methods, each tree depends on the previous one, and parallelizing is hence more difficult. But, at a lower level, all algorithms could take advantage of the parallelization of the computation of a MWST.

Overall, the PMBCL method appears as the most appealing method; it always combines fast convergence (as fast as bagging) when the number of terms of the mixture is increased and, from the computational point of view, it is also the most efficient one among those that we investigated, about 20-30 times faster than bagging in realistic conditions; furthermore it is easy to parallelize. Neverteless, the inertial heuristic with warm start is competitive as well.

Table 2. Serial minimum computing times (given for $n = 200$ variables)

Method	Complexity	running time (500 trees - except CL)		
		200 samples	600 samples	1000 samples
Chow-Liu	$n^2 \log(n)$	1	3.07	5.3
Bagged CL Trees	$mn^2 \log(n)$	532	1531	2674
ISH	$mn \log(n)$	45	186	432
PMBCL	$n^2 + mK(\alpha) \log(K(\alpha))$	21	82	191
Warm Start ISH	$n^2 \log(n) + mn \log(n)$	45	192	406

Table 3. Serial minimum computing times (given for $n = 1000$ variables)

Method	Complexity	running time (100 trees - except CL)		
		200 samples	600 samples	1000 samples
Chow-Liu	$n^2 \log(n)$	37	98	174
Bagged CL Trees	$mn^2 \log(n)$	5037	11662	19431
ISH	$mn \log(n)$	181	800	1433
PMBCL	$n^2 + mK(\alpha) \log(K(\alpha))$	139	612	1005
Warm Start ISH	$n^2 \log(n) + mn \log(n)$	218	766	1359

Note that convergence speed may vary between methods, and some might require fewer iterations before performance (almost) stabilizes.

6 Conclusion

In this paper we have studied variance reduction oriented model averaging techniques for density estimation, using probabilistic graphical models and more precisely mixtures of Markov trees. Those models are particularly suited for problems defined on very high dimensional spaces due to their scalability.

The contributions of this paper are the proposal of algorithms for learning mixtures of Markov trees designed to approach the quality of approximation of mixtures of bagged Chow-Liu trees at a lower computational cost, and the study of their main properties. The bottleneck of the baseline bagging method is the quadratic number of edges considered for building the structure of each Markov tree of the ensemble. This is problematic since it may lead to restricting the total number of trees in the mixture, while on the other hand larger numbers of trees would yield more accurate models. The main idea behind our proposals is to use the information obtained from the computation of a first tree of the mixture so as to simplify the computation of the subsequent trees of the mixture.

We have demonstrated on synthetic datasets the interest of Markov trees averaging over regularizing a single Chow-Liu tree. For example, when enough Bagged Chow-Liu trees are averaged, they outperform that single model. Likewise, we have shown that our approximation schemes match the accuracy of bagging better than existing alternatives. Among the proposed methods, the most robust and computationally efficient one (PMBCL) defines a set of candidate edges by selecting all edges computed at the first iteration that are better

than a constant complexity "edge penalty", and subsequently only considers those edges for building remaining ensemble terms. The approximation schemes that we have proposed were in our experiments one order of magnitude faster than Mixtures of Bagged Chow-Liu Trees.

Other variants of our methods could be investigated in the future. For example, it might be interesting to perform a looser selection of edges at the first iteration, and to include additional regularization when learning subsequent terms, or vice versa. This might further ease the calibration of the tradeoff between computational complexity gains and variance reduction potential.

Acknowledgments. François Schnitzler is supported by a F.R.I.A. scholarship. Pierre Geurts is a research associate of the FNRS, Belgium. This work was also funded by the Biomagnet IUAP network of the Belgian Science Policy Office and the Pascal2 network of excellence of the EC. The scientific responsibility is the authors'.

References

1. Aliferis, C., Statnikov, A., Tsamardinos, I., Mani, S., Koutsoukos, X.: Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. JMLR 11, 171–234 (2010)
2. Ammar, S., Leray, P., Defourny, B., Wehenkel, L.: Probability density estimation by perturbing and combining tree structured Markov networks. In: Sossai, C., Chemello, G. (eds.) ECSQARU 2009. LNCS, vol. 5590, pp. 156–167. Springer, Heidelberg (2009)
3. Ammar, S., Leray, P., Schnitzler, F., Wehenkel, L.: Sub-quadratic Markov tree mixture learning based on randomizations of the Chow-Liu algorithm. In: The Fifth European Workshop on Probabilistic Graphical Models, pp. 17–24 (2010)
4. Ammar, S., Leray, P., Wehenkel, L.: Sub-quadratic Markov tree mixture models for probability density estimation. In: 19th International Conference on Computational Statistics (COMP-STAT 2010), pp. 673–680 (2010)
5. Auvray, V., Wehenkel, L.: On the construction of the inclusion boundary neighbourhood for Markov equivalence classes of bayesian network structures. In: Proceedings of 18th Conference on Uncertainty in Artificial Intelligence, pp. 26–35 (2002)
6. Auvray, V., Wehenkel, L.: Learning inclusion-optimal chordal graphs. In: Proceedings of 24th Conference on Uncertainty in Artificial Intelligence, pp. 18–25 (2008)
7. Bach, F.R., Jordan, M.I.: Thin junction trees. In: Advances in Neural Information Processing Systems, vol. 14, pp. 569–576. MIT Press, Cambridge (2001)
8. Breiman, L.: Arcing classifiers. Tech. rep., Dept. of Statistics, University of California (1996)
9. Chazelle, B.: A minimum spanning tree algorithm with inverse-Ackermann type complexity. J. ACM 47(6), 1028–1047 (2000)
10. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Trans. Inf. Theory 14, 462–467 (1968)
11. Cooper, G.: The computational complexity of probabilistic inference using bayesian belief networks. Artificial Intelligence 42(2-3), 393–405 (1990)
12. Efron, B., Tibshirani, R.: An introduction to the bootstrap. Chapman & Hall, Boca Raton (1993)

13. Elidan, G.: Bagged structure learning of bayesian network. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (2011)
14. Elidan, G., Gould, S.: Learning bounded treewidth bayesian networks. JMLR 9, 2699–2731 (2008)
15. Friedman, N., Goldszmidt, M., Wyner, A.: Data analysis with bayesian networks: A bootstrap approach. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 196–205. (1999)
16. Friedman, N., Nachman, I., Peér, D.: Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 206–215 (1999)
17. Kirshner, S., Smyth, P.: Infinite mixtures of trees. In: ICML 2007: Proceedings of the 24th International Conference on Machine Learning, pp. 417–423. ACM, New York (2007)
18. Koller, D., Friedman, N.: Probabilistic Graphical Models. MIT Press, Cambridge (2009)
19. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society 7(1), 48–50 (1956)
20. Kullback, S., Leibler, R.: On information and sufficiency. Ann. Math. Stat. 22(1), 79–86 (1951)
21. Kumar, M.P., Koller, D.: Learning a small mixture of trees. In: Advances in Neural Information Processing Systems, vol. 22, pp. 1051–1059 (2009)
22. Kwisthout, J.H., Bodlaender, H.L., van der Gaag, L.: The necessity of bounded treewidth for efficient inference in bayesian networks. In: the 19th European Conference on Artificial Intelligence, pp. 623–626 (2010)
23. Liu, H., Xu, M., Haijie Gu, A.G., Lafferty, J., Wasserman, L.: Forest density estimation. JMLR 12, 907–951 (2011)
24. Madigan, D., Raftery, A., Wermuth, N., York, J., Zucchini, W.: Model Selection and Accounting for Model Uncertainty in Graphical Models Using Occam's Window. Journal of the American Statistical Association 89, 1535–1546 (1994)
25. Meila, M., Jordan, M.: Learning with mixtures of trees. JMLR 1, 1–48 (2001)
26. Meila, M., Jaakkola, T.: Tractable bayesian learning of tree belief networks. In: Proc. of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pp. 380–388. Morgan Kaufmann, San Francisco (2000)
27. Ormoneit, D., Tresp, V.: Improved gaussian mixture density estimates using bayesian penalty terms and network averaging. In: Advances in Neural Information Processing Systems, pp. 542–548. MIT Press, Cambridge (1995)
28. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, San Francisco (1988)
29. Robinson, R.W.: Counting unlabeled acyclic digraphs, vol. 622. Springer, Heidelberg (1977)
30. Schnitzler, F., Leray, P., Wehenkel, L.: Towards sub-quadratic learning of probability density models in the form of mixtures of trees. In: 18th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2010), Bruges, Belgium, pp. 219–224 (2010)
31. Shahaf, D., Chechetka, A., Guestrin, C.: Learning thin junction trees via graph cuts. In: Artificial Intelligence and Statistics (AISTATS), pp. 113–120 (2009)
32. Tarjan, R.E.: Data structures and network algorithms. Society for Industrial and Applied Mathematics, Philadelphia (1983)
33. Wehenkel, L.: Decision tree pruning using an additive information quality measure. In: Uncertainty in Intelligent Systems, pp. 397–411 (1993)