Probability density estimation by perturbing and combining tree structured Markov networks

Sourour Ammar[1], Philippe Leray[1], Boris Defourny[2], and Louis Wehenkel[2]

[1] Knowledge and Decision Team
Laboratoire d'Informatique de Nantes Atlantique (LINA) UMR 6241
Ecole Polytechnique de l'Université de Nantes, France
sourour.ammar@etu.univ-nantes.fr, philippe.leray@univ-nantes.fr
[2] Department of Electrical Engineering and Computer Science & GIGA-Research,
University of Liège, Belgium
boris.defourny@ulg.ac.be, L.Wehenkel@ulg.ac.be

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, or by pure random sampling. We empirically assess the performances of these methods in terms of accuracy, with respect to mixture models derived by EM-based learning of Naive Bayes models, and EM-based learning of mixtures of trees. We find that the bagged ensembles outperform all other methods while the random ones perform also very well. Since the computational complexity of the former is quadratic and that of the latter is linear in the number of variables of interest, this paves the way towards the design of efficient density estimation methods that may be applied to problems with very large numbers of variables and comparatively very small sample sizes.

1 Introduction

Learning of graphical probabilistic models essentially aims at discovering a maximal factorization of the joint density of a set of random variables according to a graph structure, based on a random sample of joint observations of these variables [1]. Such a graphical probabilistic model may be used for elucidating the conditional independencies holding in the data-generating distribution, for automatic reasoning under uncertainties, and for Monte-Carlo simulations. Unfortunately, currently available optimization algorithms for graphical model structure learning are either restrictive in the kind of distributions they search for, or of too high computational complexity to be applicable in very high dimensional spaces [2]. Moreover, not much is known about the behavior of these methods in small sample conditions and, as a matter of fact, one may suspect that they will suffer from overfitting when the number of variables is very large and the sample size is comparatively very small.

In the context of supervised learning, a generic framework which has led to many fruitful innovations is called "Perturb and Combine". Its main idea is to

on the one hand perturb in different ways the optimization algorithm used to derive a predictor from a data-set and on the other hand to combine in some appropriate fashion a set of predictors obtained by multiple iterations of the perturbed algorithm over the data-set. In this framework, ensembles of weakly fitted randomized models have been studied intensively and used successfully during the last two decades. Among the advantages of these methods, let us quote the improved predictive accuracy of their models, and the potentially improved scalability of their learning algorithms. For example, ensembles of bagged (derived from bootstrap copies of the dataset) or extremely randomized decision or regression trees, as well as random forests, have been applied successfully in complex high-dimensional tasks, as image and sequence classification [3].

In the context of density estimation, bagging (and boosting) of normal distributions has been proposed by Ridgeway [4]. In [5] the Perturb and Combine idea for probability density estimation with probabilistic graphical models was first explored by comparing large ensembles of randomly generated (directed) poly-trees and randomly generated undirected trees. One of the main findings of that work is that poly-trees, although more expressive, do not yield more accurate ensemble models in this context than undirected trees.

Thus, in the present paper we focus on tree structured undirected probabilistic graphical networks (we will call them Markov tree mixtures in the sequel) and we study various randomization and averaging schemes for generating such models. We consider two simple and in some sense extreme instances of this class of methods, namely ensembles of optimal trees derived from bootstrap copies of the dataset by the Chow and Liu algorithm [6], which is of quadratic complexity with respect to the number of variables (we call this bagging of Markov trees), and mixtures of tree structures generated in a totally randomized fashion with linear complexity in the number of variables (we will call them totally randomized Markov tree mixtures). We assess the accuracy of these two methods empirically on a set of synthetic test problems in comparison to EM-based state of the art methods building respectively Naive Bayes models and mixtures of trees, as well as a golden standard which uses the structure of the target distribution.

The rest of this paper is organized as follows. Section 2 recalls the classical Bayesian framework for learning mixtures of models and Section 3 describes the proposed algorithms. Section 4 collects our simulation results, Section 5 discusses the main findings of our work, and Section 6 briefly concludes and highlights some directions for further research.

2 Bayesian modeling framework

Let $X = \{X_1, \ldots, X_n\}$ be a finite set of discrete random variables, and $D = (x^1, \cdots, x^d)$ be a data-set (sample) of joint observations $x^i = (x_1^i, \cdots, x_n^i)$ independently drawn from some data-generating density $\mathbb{P}_G(X)$.

In the full Bayesian approach, one assumes that $\mathbb{P}_G(X)$ belongs to some space of densities $\mathcal{D}$ described by a model-*structure* $M \in \mathcal{M}$ and model-*parameters* $\theta_M \in \Theta_M$, and one infers from the data-set a mixture of models described by

the following equation:

$$\mathbb{P}_{\mathcal{D}}(X|D) = \sum_{M \in \mathcal{M}} \mathbb{P}(M|D)\ \mathbb{P}(X|M, D), \tag{1}$$

where $\mathbb{P}(M|D)$ is the posterior probability over the model-space $\mathcal{M}$ conditionally to the data D, and where $\mathbb{P}(X|M, D)$ is the integral:

$$\mathbb{P}(X|M, D) = \int_{\Theta_M} \mathbb{P}(X|\theta_M, M)\ d\mathbb{P}(\theta_M|M, D). \tag{2}$$

So $\mathbb{P}_{\mathcal{D}}(X|D)$ is computed by:

$$\mathbb{P}_{\mathcal{D}}(X|D) = \sum_{M \in \mathcal{M}} \mathbb{P}(M|D) \int_{\Theta_M} \mathbb{P}(X|\theta_M, M)\ d\mathbb{P}(\theta_M|M, D), \tag{3}$$

where $d\mathbb{P}(\theta_M|M, D)$ is the posterior model-parameter density and $\mathbb{P}(X|\theta_M, M)$ is the likelihood of observation X for the structure M with parameters θ_M.

When the space of model-structures $\mathcal{M}$ and corresponding model-parameter Θ_M is the space of Bayesian networks or the space of Markov networks over X, approximations have to be done in order to make tractable the computation of equation (3). For Bayesian networks for example, it shown in [7] that equation (2) can be simplified by the likelihood estimated with the parameters of maximum a posteriori probability (MAP) $\tilde{\theta}_M = \arg\max_{\theta_M} \mathbb{P}(\theta_M|M, D)$, under the assumption of a Dirichlet distribution (parametrized by its coefficients α_i) for the prior distribution of the parameters $\mathbb{P}(\theta_M|M)$.

Another approximation to consider is simplifying the summation over all the possible model-structures M. As the size of the set of possible graphical model structures is super-exponential in the number of variables [8], the summation of equation (1) must in practice be performed over a strongly constrained subspace $\hat{\mathcal{M}}$ obtained for instance by sampling methods [9–11], yielding the approximation

$$\mathbb{P}_{\hat{\mathcal{M}}}(X|D) = \sum_{M \in \hat{\mathcal{M}}} \mathbb{P}(M|D)\mathbb{P}(X|\tilde{\theta}_M, M). \tag{4}$$

Let's note here that this equation is simplified once more when using classical structure learning methods, by keeping only the model $M = \tilde{M}$ maximizing $\mathbb{P}(M|D)$ over $\mathcal{M}$:

$$\mathbb{P}_{\tilde{M}}(X|D) = \mathbb{P}(X|\tilde{\theta}_{\tilde{M}}, \tilde{M}). \tag{5}$$

3 Randomized Markov tree mixtures

In this work, we propose to choose as set $\hat{\mathcal{M}}$ in equation (4) a randomly generated subset of pre-specified cardinality of Markov tree models.

3.1 Poly-tree models

A poly-tree model for the density over X is defined by a Directed Acyclic Graph (DAG) structure P which skeleton is acyclic and connected, and the set of vertices of which is in bijection with $X = \{X_1, \ldots, X_n\}$, together with a set of conditional densities $\mathbb{P}_P(X_i|pa_P(X_i))$, where $pa_P(X_i)$ denotes the set of variables in bijection with the parents of X_i in P. Like more general DAGs, this structure P represents graphically the density factorization

$$\mathbb{P}_P(X) = \prod_{i=1}^{n} \mathbb{P}_P(X_i|pa_P(X_i)). \tag{6}$$

The model parameters are thus here specified by the vector of conditional distributions:

$$\theta_P = (\mathbb{P}_P(X_i|pa_P(X_i)))_{i=1}^{n}\,. \tag{7}$$

The structure P can be exploited for probabilistic inference over $\mathbb{P}_P(X)$ with a computational complexity linear in the number of variables n [12].

One can define nested subclasses $\mathcal{P}^p$ of poly-tree structures by imposing constraints on the maximum number p of parents of any node. In these subclasses, not only inference but also parameter learning is of linear complexity in the number of variables.

3.2 Markov tree models

The smallest subclass of poly-tree structures is called the Markov tree subspace, in which nodes have exactly one parent ($p = 1$). Markov tree models have the essential property of having no v-structures [1], in addition to the fact that their skeleton is a tree, and their dependency model may be read-off without taking into account the direction of their arcs. In other words, a poly-tree model without v-structures is a Markov tree and is essentially defined by its skeleton. These are the kind of models that we will consider subsequently in this paper. Importantly, Markov tree models may be learned efficiently by the Chow and Liu algorithm which is only quadratic in the number of vertices (variables) [6]. Given the skeleton of the Markov tree, one can derive an equivalent directed acyclic (poly-tree) graph from it by arbitrarily choosing a root node and by orienting the arcs outwards from this node in a recursive fashion.

3.3 Mixtures of Markov trees

A mixture distribution $\mathbb{P}_{\hat{\mathcal{T}}}(X_1, \ldots, X_n)$ over a set $\hat{\mathcal{T}} = \{T_1, \ldots, T_m\}$ of m Markov trees is defined as a convex combination of elementary Markov tree densities, ie.

$$\mathbb{P}_{\hat{\mathcal{T}}}(X) = \sum_{i=1}^{m} \mu_i \mathbb{P}_{T_i}(X), \tag{8}$$

where $\mu_i \in [0,1]$ and $\sum_{i=1}^{m} \mu_i = 1$, and where we leave for the sake of simplicity implicit the values of the parameter sets $\tilde{\theta}_i$ of the individual Markov tree densities.

While single Markov tree models impose strong restrictions on the kind of densities they can faithfully represent, mixtures of Markov trees, as well as mixtures of empty graphs (i.e. Naive Bayes with hidden class), are universal approximators (see, e.g., [13]).

3.4 Generic randomized Markov tree mixture learning algorithm

Our generic procedure for learning a random Markov tree mixture distribution from a data-set D is described by Algorithm 1; it receives as inputs X, D, m, and three procedures *DrawMarkovtree*, *LearnPars*, *CompWeights*.

Algorithm 1 (Learning a Markov tree mixture)

1. *Repeat for* $i = 1, \cdots, m$*:*
   (a) Draw random number ρ_i*,*
   (b) $T_i = DrawMarkovtree(D, \rho_i)$*,*
   (c) $\tilde{\theta}_{T_i} = LearnPars(T_i, D, \rho_i)$
2. $(\mu)_{i=1}^{m} = CompWeights((T_i, \theta_{T_i}, \rho_i)_{i=1}^{m}, D)$
3. *Return* $\left(\mu_i, T_i, \tilde{\theta}_{T_i}\right)_{i=1}^{m}$.

Line (a) of this algorithm draws a random number in $\rho_i \in [0;1)$ uniformly disributed, which may be used as a seed by the two subsequently used by randomizations of *DrawMarkovtree* which builds a tree structure T_i possibly depending on the dataset D and ρ_i and *LearnPars* which estimates the parameters of T_i. Versions of these two procedures used in our experiments are further discussed in the next section. The algorithm returns the m tree-models, along with their parameters θ_{T_i} and the weights of the trees μ_i.

3.5 Specific variants

In our investigations reported below, we have decided to compare various versions of the above generic algorithm.

In particular, we consider two variants of the *DrawMarkovtree* function: one that randomly generates unconstrained Markov trees (by sampling from a uniform density over the set $\mathcal{P}^1$ of all Markov tree models, see [5] for details), and one that builds optimal tree structures by applying the MWST (Maximum Weight Spanning Tree) structure learning algorithm published in the late sixties [6] by Chow and Liu on a random bootstrap [14] replica of the initial learning set D. The random sampling procedure of the first variant is described in [5]. The second variant reminds the Bagging idea of [15].

Concerning the parameter estimation by *LearnPars*, we use the BDeu score maximization for each tree structure individually, which is tantamount to selecting the MAP estimates using Dirichlet priors. More specifically, in our experiments which are limited to binary random variables, we used non-informative

priors, which then amounts to using $\alpha = 1/2$, i.e. $p(\theta, 1-\theta) \propto \theta^{-1/2}(1-\theta)^{-1/2}$ for the prior density of the parameters characterizing the conditional densities attached the Markov tree nodes, once this tree is oriented in an arbitrary fashion. Notice that in the case of tree-bagging, these parameters are estimated from the bootstrap sample used to generate the corresponding tree structure.

Finally, we consider two variants for *CompWeights* function, namely uniform weighting (where coefficients are defined by $\mu_i = \frac{1}{m}, \forall i = 1, \ldots, m$) and Bayesian averaging (where coefficients μ_i are proportional to the posterior probability of the Markov tree structure T_i, derived from its BDeu score [1] computed from the full data-set D).

4 Empirical simulations

4.1 Protocol

In order to evaluate the four different variants of our algorithm, we carried out repetitive experiments for different data-generating (or target) densities, by proceeding in the following way.

Choice of target density All our experiments were carried out with models for a set of 8 and 16 binary random variables. We chose to start our investigations in such a simple setting in order to be able to compute accuracies exactly (see Section 4.1), and so that we can easily analyze the graphical structures of the target densities and of the inferred set of poly-trees.

To choose a target density $\mathbb{P}_G(X)$, we first decide whether it will factorize according to a poly-tree or to a more general directed acyclic graph structure. Then we use the appropriate random structure generation algorithm described in [5] to draw a structure and, we choose the parameters of the target density by selecting for each conditional density of the structure (they are all related to binary variables) two random numbers in the interval $[0, 1]$ and by normalizing.

Generation of data-sets For each target density and data-set size, we generated 10 different data-sets by sampling values of the random variables using the Monte-Carlo method with the target structure and parameter values.

We carried out simulations with data-set sizes of N = 250 elements for models with 8 or 16 variables and for N= 2000 for the models with 16 variables. Given the total number of 2^n possible configurations of our n random variables, we thus look at rather small data-sets.

Learning of mixtures For a given data-set and for a given variant of the mixture learning algorithm we generate ensemble models of growing sizes, respectively $m = 1$, $m = 10$, and then up to $m = 500$ by increments of 10. This allows us to appraise the effect of the ensemble size on the quality of the resulting model.

Accuracy evaluation The quality of any density inferred from a data-set is evaluated by the (asymmetric) Kullback-Leibler divergence [16] between this density and the data-generating density $\mathbb{P}_G(X)$ used to generate the data-set. This is exactly computed by

$$KL(\mathbb{P}_G, \mathbb{P}_M) = \sum_{X \in \mathcal{X}} \mathbb{P}_G(X) \ln \frac{\mathbb{P}_G(X)}{\mathbb{P}_M(X)}, \tag{9}$$

where $\mathbb{P}_M(X)$ denotes the density that is evaluated, and $\mathcal{X}$ denotes the set of all possible configurations of the random variables in X.

Reference methods We also provide comparative accuracy values obtained in the same fashion with five different reference methods, namely (i) a *golden standard* denoted by GO which is obtained by using the data-generating structure and reestimating its parameters from the dataset D, (ii) a series of *Naive Bayes* models with growing number of hidden classes denoted by NBE^* and built according to the Expectation-Maximization (EM) algorithm [17] as proposed in [18] but without pruning, (iii) a series of *Optimal Tree Mixtures* with of growing numbers of terms denoted by MT and built according to the algorithm proposed by Meila-Predoviciu which combines the Chow and Liu MWST algorithm with the EM algorithm for parameter estimation [13], (iv) a baseline method denoted by BL which uses a complete (fully connected) DAG structure which parameters are estimated from the dataset D, and (v) a single Markov tree built using the Chow and Liu algorithm on the whole dataset (denoted by CL, below).

Software implementation Our various algorithms of model generation were implemented in C++ with the Boost library [3] and various APIs provided by the ProBT© platform [4].

4.2 Results

Figure 1 (resp. Figure 2) provides a representative set of learning curves for a target density corresponding to a poly-tree distribution (resp. DAG distribution). The horizontal axis corresponds to the number m of mixture terms, whereas the vertical axis corresponds to the KL measures with respect to the target density. All the curves represent average results obtained over ten different data-sets of 250 learning samples and five target distributions. Before analyzing these curves, let us first remind that in our first experiments reported in [5], which compared mixtures of fully randomly generated poly-trees with mixtures of fully randomly generated Markov trees, we found that general poly-tree mixtures were not significantly different from Markov tree mixtures in terms of their accuracies. Thus we have decided in the present paper to report only results obtained with Markov tree mixtures and a broader set of reference methods.

[3] `http://www.boost.org/`

[4] `http://bayesian-programming.org`

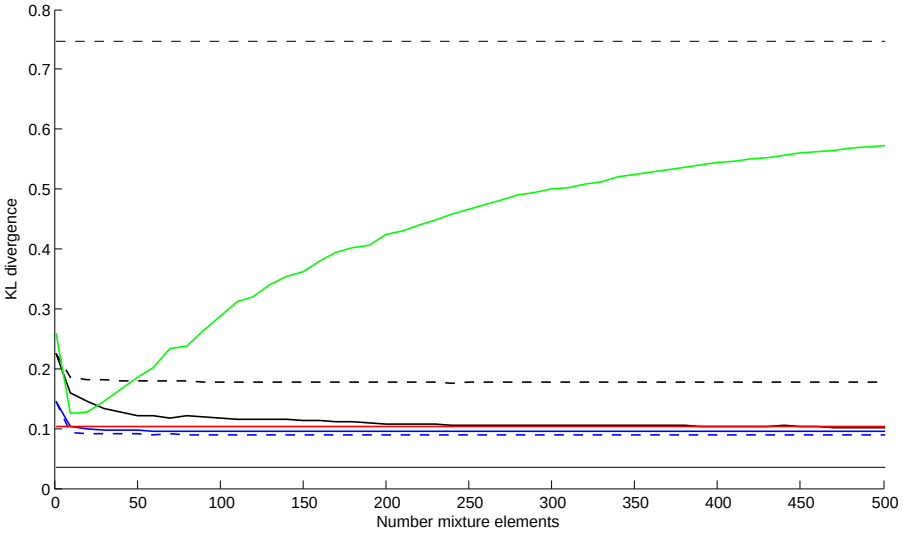

Fig. 1. Example results. Mixtures of trees for density estimation with a poly-tree target distribution. 10 experiments with a sample size of 250 for 5 random target distributions of 8 variables. (lower is better; see text for explanation of curves legends).

The plain black horizontal line in the lower part (respectively at about $KL = 0.03$ in Fig. 1 and at $KL = 0.04$ in Fig. 2) of these graphics correspond to the golden standard GO, while the dashed black horizontal line in the upper part (respectively at $KL = 0.75$ and $KL = 1.3$) corresponds to the baseline method BL. The plain red horizotal line (around $KL = 0.1$ and $KL = 0.16$) corresponds to method CL, i.e. a single Markov tree built using the Chow and Liu algorithm on the whole training set D. The green plain (non-monotonic) curve, in the middle part of the figures, corresponds to the NBE* algorithm.

The dashed black monotonically decreasing curve in the lower part of the diagrams corresponds to uniform mixtures of totally randomly generated trees, while the black plain curve starting from the same initial point but then located below corresponds to the same mixtures when the terms are weighted according to their posterior probabilities given the dataset. The blue curves, located below, respectively dashed and plain, correspond to mixtures of bagged trees with respectively uniform and posterior probability weighting.

We thus first observe that our four random Markov tree mixture methods are clearly outperforming the baseline BL in terms of accuracy and some of them are already quite close to the golden standard GO. All four variants also nicely behave in a monotonic fashion: the more terms in the mixture the more accurate the resulting model.

Concerning the totally randomly generated tree mixtures, we also observe that when they are weighted proportionally to their posterior probability given the dataset they provide much better performances as compared to a uniform

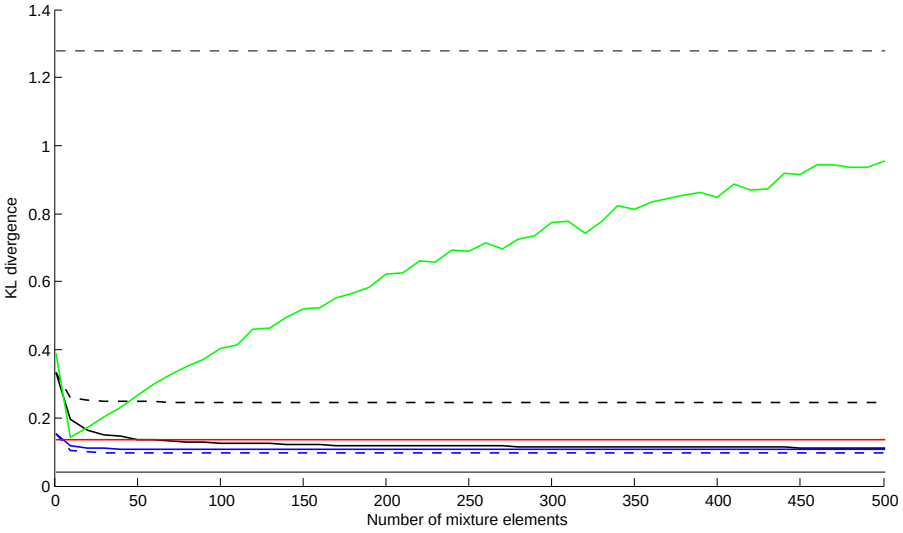

Fig. 2. Example results. Mixtures of trees for density estimation with a DAG target distribution. 10 experiments with a sample size of 250 for 5 random target distributions of 8 variables. (lower is better; see text for explanation of curves legends).

weighting procedure. Concerning the mixtures of bagged trees we observe from Figs. 1 and 2 that they both outperform the mixtures of randomly generated trees in terms of asymptotic (with respect to the number of mixture terms) accuracy and even more in terms of speed of convergence. With this bagging approach, we also notice that the uniform weighting procedure is actually slightly better than the one using weights based on the posterior probabilities given the dataset. We believe that non-uniform weighting is counterproductive in the case of bagged ensembles because it increases the variance of the ensemble model without decreasing its bias. Finally, we note that both bagging methods provide consistently better results than single CL trees built on the whole dataset, as soon as the number mixture terms is larger than about ten; this superiority is stronger in the case of DAG target distributions (Fig. 2) than for poly-tree target distributions (Fig. 1).

The NBE^* algorithm obtains results very close but slightly less good than those of bagged tree mixtures for a very small number of components (hidden classes). However, contrary to random trees or bagged trees, in the Naive Bayes method one clearly observes the fact that adding new components in the mixture eventually, and rather quickly, leads to strong overfitting.

All in all, the consistently best method in these trials is the method which uses bagging of tree structures combined with a uniform weighting scheme.

Note to the reviewers:

At the time of submitting this paper, the experiments with the MT algorithm are still in progress (they will be reported in the final version of the paper, if it is

accepted). Our first observations show that they provide similar results (although slightly better) than those of Naive Bayes, namely for a small number of terms they yield accuracies comparable to the bagged tree mixtures, but then when the number mixture terms increases they also lead to overfitting.

Experiments for 16 variables and 250 or 2000 sample size are also in progress and will be reported in the final version of the paper.

In the final version, the graphics of Figs. 1 and 2 will also be improved to ensure a better legibility on a printed black and white version of the paper.

5 Discussion

Our simulation results showed that in small sample conditions the mixtures of Markov trees turned out to be quite often of comparable accuracy than the golden standard GO, and in general largely superior to the complete structure baseline BL.

Bagged ensembles of Markov trees significantly outperform totally randomized ensembles of Markov trees, both in terms of accuracy and in terms of speed of convergence when the number of mixture components is increased. Contrary to the more sophisticated EM-based Naive Bayes and Mixtree methods, our methods do not lead to overfitting when the number of mixture terms is increased.

From a computational point of view, bagging which uses the Chow Liu MWST algorithm as baselearner is quadratic in the number of variables while the generation of random tree structures may be done in linear time (see [5]).

In between these two extreme randomization schemes, one can imagine a whole range of methods based on the combination of bootstrap resampling and more or less strong randomization of the generation of the Markov trees leading to different computational trade-offs. Also, out-of-bag estimates may be exploited to compute unbiased accuracies of the ensemble models [4].

6 Summary and future works

We have investigated in this paper the transposition of the "Perturb and Combine" idea celebrated in supervised learning to the context of unsupervised density estimation with graphical probabilistic models. We have presented a generic framework for doing this, based on randomized mixtures of tree structured Markov networks, where the perturbation was done by generating in a totally random fashion the structure component, or by bootstrapping data before optimizing this structure component.

The first results obtained in the context of a simple test protocol are already very promising, while they also highlight a certain number of immediate future research directions.

Thus, a first line of research will be to apply our experimental protocol to a larger set of problems including high-dimensional ones and a larger range of sample sizes. We believe also that a more in depth analysis of the accuracy

results with respect to the basic properties of the target distributions as well as sample sizes would be of interest, in particular with the aim of characterizing more precisely under which conditions our methods are more effective than state-of-the-art ones. Of course, these investigations should also aim at systematically comparing all these algorithm variants from the point of view of their a computational complexity.

Another more generic direction of research, is to adapt importance sampling approaches (e.g. the cross-entropy method [19]) in order to generate randomized ensembles of simple structures (chains, trees, poly-trees, etc.) that fit well the given data-set.

While the class of methods investigated in this paper is based on generating an ensemble by drawing its terms from a same distribution (which could be done in parallel), we believe that the combination of these method with sequential methods such as Boosting or Markov-Chain Monte-Carlo which have already been applied in the context of graphical probabilistic models (see e.g. [20]), might provide a very rich avenue for the design of novel density estimation algorithms.

Acknowledgments

This paper presents research results of the Belgian Network BIOMAGNET (Bioinformatics and Modeling: from Genomes to Networks), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The authors thank Pierre Geurts and François Schnitzler for useful discussions and suggestions to improve the manuscript.

References

1. Cowell, R., Dawid, A., Lauritzen, S., Spiegelhalter, D.: Probabilistic Networks and Expert Systems. Springer (1999)
2. Auvray, V., Wehenkel, L.: Learning inclusion-optimal chordal graphs. In: Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI-08), Morgan Kaufmann Publishers (2008) 18–25
3. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning **63**(1) (2006) 3–42
4. Ridgeway, G.: Looking for lumps: boosting and bagging for density estimation. Computational Statistics & Data Analysis **38**(4) (2002) 379 – 392
5. Ammar, S., Leray, P., Defourny, B., Wehenkel, L.: High-dimensional probability density estimation with randomized ensembles of tree structured bayesian networks. In: Proceedings of the fourth European Workshop on Probabilistic Graphical Models (PGM08). (2008) 9–16
6. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory **14**(3) (1968) 462–467
7. Chickering, D., Heckerman, D.: Efficient approximations for the marginal likelihood of bayesian networks with hidden variables. Machine Learning **29**(2-3) (1997) 181–212
8. Robinson, R.: Counting unlabeled acyclic digraphs. In Little, C.H.C., ed.: Combinatorial Mathematics V. Volume 622 of Lecture Notes in Mathematics., Berlin, Springer (1977) 28–43

9. Madigan, D., Raftery, A.: Model selection and accounting for model uncertainty in graphical models using occam's window. The American Statistical Association **89** (1994) 1535–1546
10. Madigan, D., York, J.: Bayesian graphical models for discrete data. International Statistical Review **63** (1995) 215–232
11. Friedman, N., Koller, D.: Being bayesian about network structure. In Boutilier, C., Goldszmidt, M., eds.: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-00), Morgan Kaufmann Publishers (2000) 201–210
12. Pearl, J.: Fusion, propagation, and structuring in belief networks. Artificial Intelligence **29** (1986) 241–288
13. Meila-Predoviciu, M.: Learning with Mixtures of Trees. PhD thesis, MIT (1999)
14. Efron, B., Tibshirani, R.J.: An introduction to the Bootstrap. Volume 57 of Monographs on Statistics and Applied Probability. Chapman and Hall (1993)
15. Breiman, L.: Bagging predictors. Machine Learning **24**(2) (1996) 123–140
16. Kullback, S., Leibler, R.: On information and sufficiency. Annals of Mathematical Statistics **22**(1) (1951) 79–86
17. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological) **39**(1) (1977) 1–38
18. Lowd, D., Domingos, P.: Naive bayes models for probability estimation. In: (ICML '05) Proceedings of the 22nd international conference on Machine Learning, ACM (2005) 529–536
19. Rubinstein, R., Kroese, D.: The Cross-Entropy Method. A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning. Information Science and Statistics. Springer (2004)
20. Rosset, S., Segal, E.: Boosting density estimation. In: Proceedings of the 16th International Conference on Neural Information Processing Systems (NIPS), Vancouver, British Columbia, Canada (2002) 267–281