High-dimensional probability density estimation with randomized ensembles of tree structured Bayesian networks

Sourour Ammar and Philippe Leray
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

Boris Defourny and Louis Wehenkel
Department of Electrical Engineering and Computer Science & GIGA-Research,
University of Liège, Belgium
boris.defourny@ulg.ac.be, L.Wehenkel@ulg.ac.be

Abstract

In this work we explore the Perturb and Combine idea, celebrated in supervised learning, in the context of probability density estimation in high-dimensional spaces with graphical probabilistic models. We propose a new family of unsupervised learning methods of mixtures of large ensembles of randomly generated tree or poly-tree structures. The specific feature of these methods is their scalability to very large numbers of variables and training instances. We explore various simple variants of these methods empirically on a set of discrete test problems of growing complexity.

1 Introduction

Learning of Bayesian networks aims at modeling the joint density of a set of random variables from a random sample of joint observations of these variables (Cowell et al., 1999). Such a graphical model may be used for elucidating the conditional independences holding in the data-generating distribution, for automatic reasoning under uncertainties, and for Monte-Carlo simulations. Unfortunately, currently available algorithms for Bayesian network 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 (Auvray and Wehenkel, 2008).

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 scalability of their learning algorithms and the improved predictive accuracy of their models. For example, ensembles of extremely randomized trees have been applied successfully in complex high-dimensional tasks, as image and sequence classification (Geurts et al., 2006).

In this work we explore the Perturb and Combine idea for probability density estimation. We study a family of learning methods to infer mixtures of large ensembles of randomly generated tree structured Bayesian networks. The specific feature of these methods is their scalability to very large numbers of variables and training instances. We explore various simple variants of these methods empirically on a set of discrete test problems of growing complexity.

The rest of this paper is organized as follows.

Section 2 discusses the classical Bayesian framework for learning mixtures of models. Section 3 describes the proposed approach and algorithms presented in this paper and Section 4 reports simulation results on a class of simple discrete test problems. Section 5 discusses our work in relation with the literature 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 $x^i = (x^i_1, \cdots, x^i_n)$ 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:

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

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

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

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

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

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

When the space of model-structures $\mathcal{M}$ is the space of Bayesian networks over X, approximations have to be done in order to make tractable the computation of Equation (3). (Chickering and Heckerman, 1997) show that Equation (2) can be simplified by the likelihood estimated with the parameters of maximum posterior probability $\tilde{\theta}_M = \arg\max_{\theta_M} \mathbb{P}(\theta_M|M, D)$, under the assumption of a Dirichlet distribution (parametrized by its coefficients α_i) for the prior distribution of the parameters $\mathbb{P}(\theta_M)$.

Another approximation to consider is simplifying the summation over all the possible model-structures M. As the size of the set of possible Bayesian network structures is super-exponential in the number of variables (Robinson, 1977), the summation of Equation (1) must be performed over a strongly constrained subspace $\hat{\mathcal{M}}$ obtained for instance by sampling methods (Madigan and Raftery, 1994; Madigan and York, 1995; Friedman and Koller, 2000), yielding the approximation

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

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

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

Let us emphasize that this further simplication has also the advantage of producing a *single* graphical model from which one can read of independencies *directly*. This may however be at the price of a possibly significant reduction of accuracy of the density estimation.

3 Randomized poly-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 poly-tree models.

3.1 Poly-tree models

A poly-tree model for the density over X is defined by a directed acyclic graph structure P whose skeleton is acyclic and connected, and whose set of vertices is in bijection with X, 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. The structure P represents graphically the density factorization

$$\mathbb{P}_P(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbb{P}_P(X_i|pa_P(X_i)). \quad (6)$$

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

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

The structure P can be exploited for probabilistic inference over $\mathbb{P}_P(X_1, \ldots, X_n)$ with a computational complexity linear in the number of variables n (Pearl, 1986).

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. The smallest such subclass is called the tree subspace, in which nodes have exactly one parent ($p = 1$).

When necessary, we will denote by $\mathcal{P}^*$ (respectively $\mathcal{P}^1$) the space of all possible poly-tree (respectively tree) structures defined over X.

3.2 Mixtures of poly-trees

A mixture distribution $\mathbb{P}_{\hat{\mathcal{P}}}(X_1, \ldots, X_n)$ over a set $\hat{\mathcal{P}} = \{P_1, \ldots, P_m\}$ of m poly-trees is defined as a convex combination of elementary poly-tree densities, i.e.

$$\mathbb{P}_{\hat{\mathcal{P}}}(X_1, \ldots, X_n) = \sum_{i=1}^{m} \mu_i \mathbb{P}_{P_i}(X_1, \ldots, X_n), \quad (8)$$

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 poly-tree densities.

While single poly-tree models impose strong restrictions on the kind of densities they can represent, mixtures of poly-trees are universal approximators, as well as mixtures of trees or chains, or even mixtures of empty graphs (i.e. Naive Bayes with hidden class), as shown in (Meila-Predoviciu, 1999) section 3.1.

3.3 Random poly-tree mixture learning

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

Algorithm 1 (Learning a poly-tree mixture)

1. Repeat for $i = 1, \cdots, m$:
   (a) $P_i = \mathit{DrawPolytree}$,
   (b) For $j = 1, \cdots, n$:
   $\tilde{\theta}_{P_i} = \mathit{LearnPars}(P_i, D)$
2. $(\mu)_{i=1}^m = \mathit{CompWeights}((P_i, \tilde{\theta}_{P_i})_{i=1}^m, D)$
3. Return $\left(\mu_i, P_i, \tilde{\theta}_{P_i}\right)_{i=1}^m$.

3.4 Specific variants

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

In particular, we consider both mixtures of randomly generated subsets of unconstrained poly-trees (by sampling from a uniform density over $\mathcal{P}^*$), and mixtures of tree structures (by sampling from a uniform density over $\mathcal{P}^1$). The random sampling procedures are described in Section 3.5.

As concerns the mixture coefficients, we will compare two variants, namely uniform weighting (coefficient $\mu_i = \frac{1}{m}, \forall i = 1, \ldots, m$) and Bayesian averaging (coefficient μ_i proportional to the posterior probability of the poly-tree structure P_i, derived from its BDeu score computed from the data-set (Cowell et al., 1999)).

Notice that with large data-sets, the Bayesian averaging approach tends to put most of the weight on the poly-tree which has the largest score; hence to better appraise the mixture effect, we will also provide results for the model which uses only the highest score structure among the m poly-trees of the ensemble, which amounts to a kind of random search for the MAP structure defined in Equation (5).

Finally, concerning parameter estimation, we use the BDeu score maximization for each poly-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 poly-tree nodes.

3.5 Random generation of structures

Our mixture of random poly-trees and the experimental protocol described in Section 4 are based on random sampling of several classes of graphical structures (trees, poly-trees, and directed acyclic graphs).

For sampling trees and poly-trees, we chose to adapt the algorithm proposed by (Quiroz, 1989), which uses Prûfer coding of undirected tree structures. This algorithm allows to sample labelled undirected trees uniformly. We have adapted it in order to sample uniformly from the space of directed (rooted) trees and poly-trees. The resulting structure sampling algorithms are efficient, since their complexity remains linear in the number of variables. Notice however, that only in the case of tree structures these algorithms sample uniformly from the Markov equivalence classes induced by these structures. We do not know of any efficient adaptation of these algorithms to sample uniformly from structures of poly-trees with bounded number of in-degrees or to sample uniformly from Markov equivalence classes of poly-trees.

For sampling of directed acyclic graphs we used, on the other hand, the procedure given in (Ide et al., 2004), which allows to generate random structures which are of bounded in-degree and which are constrained to be connected. This scheme does neither yield a uniform sampling of these structures nor of their equivalence classes.

4 Preliminary empirical simulations

4.1 Protocol

For a first comparison of the different variants of our algorithm, we carried out repetitive experiments for different data-generating (or target) densities. All our experiments were carried out with models for a set of eight 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 4.1.4), and so that we can easily analyze the graphical structures of the target densities and of the inferred set of poly-trees.

4.1.1 Choice of target density

To choose a target density $\mathbb{P}_G(X)$, we first decide whether it will factorize according to a poly-tree or to a directed acyclic graph structure. Then we use the appropriate random structure generation algorithm (see Section 3.5) 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.

4.1.2 Generation of data-sets

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

We carry out simulations with data-set sizes of 250 and 2000 elements respectively. Given the total number of 256 possible configurations of our eight random variables, we thus look at both small and large data-sets.

4.1.3 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 $m = 1000$ by increments of 10. This allows us to appraise the effect of the ensemble size on the quality of the resulting model.

4.1.4 Accuracy evaluation

The quality of any density inferred from a data-set is evaluated by the symmetric Kulback-Leibler divergence (Kullback and Leibler, 1951) between this density and the data-generating density $\mathbb{P}_G(X)$ used to generate the data-set. This is computed by

$$KL_s(\mathbb{P}_G, \mathbb{P}_M) = KL(\mathbb{P}_G||\mathbb{P}_M) + KL(\mathbb{P}_M||\mathbb{P}_G), \quad (9)$$

where $\mathbb{P}_M(X)$ denotes the density that is evaluated, and where

$$KL(\mathbb{P}||\mathbb{P}') = \sum_{X \in \mathcal{X}} \mathbb{P}(X) \ln \left(\frac{\mathbb{P}(X)}{\mathbb{P}'(X)} \right), \quad (10)$$

and $\mathcal{X}$ denotes the set of all possible configurations of the random variables in X.

We use this formula to evaluate our mixture models, and we also provide baseline values obtained with two different reference models, namely a *baseline* approach M_0 where a complete directed acyclic model is used with parameter values inferred by BDeu score maximization on the data-set, as well as a *golden standard* M_1 where the parameters of the target structure used to generate the data-set are re-estimated by BDeu score maximization from the data-set.

4.1.5 Software implementation

Our various algorithms of model generation were implemented in C++ with the Boost library available at `http://www.boost.org/` and various APIs provided by the ProBT© platform available at `http://bayesian-programming.org`.

4.2 Results

4.2.1 Sample of results

Figure 1 provides a representative set of learning curves for a target density corresponding to the directed acyclic graph (DAG) represented on he top of the figure. The middle and lower parts represent the learning curves obtained with respectively 250 and 2000 observations in the data-set. The horizontal axis corresponds to the number m of mixture terms, whereas the vertical axis corresponds to the KL_s measures with respect to the target density. All the curves represent average results obtained over ten different data-sets of the specified size.

The dashed horizontal lines in the lower parts of these graphics correspond to the golden standard M_1, whereas the plain horizontal line (not shown on the middle graphic) correspond to the M_0 baseline (its results are very bad on the small data-set and were therefore not shown).

The dashed, respectively black and red, curves in the upper part of both diagrams correspond to uniform mixtures of, respectively trees and poly-trees. We observe that their performances are quite disappointing, even though uniform poly-tree mixtures are slightly better than uniform tree mixtures.

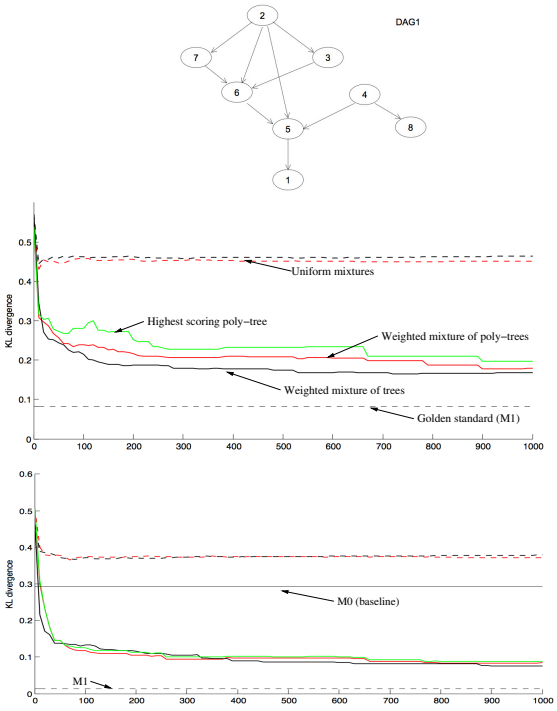

Figure 1: Example results. Top: target density DAG structure. Middle: learning curves with data-set size of 250 observation. Bottom: learning curves with 2000 observation. (see text for explanation of curves legends).

The two plain curves, respectively black and red, in the lower parts of the diagrams correspond to mixtures of respectively trees and poly-trees, when they are weighted proportionally to their posterior probability given the data-set. They provide much better performances, as compared to the baseline M_0 and are competitive with the golden standard M_1. We also observe that for the smaller sample size the tree mixtures outperform the poly-tree mixtures, whereas for the larger sample size they provide identical performances.

For the sake of comparison, we have also provided the behaviour of the "trivial mixture" (in green) which retains only the highest scoring structure of the generated ensemble. We observe that in small sample conditions, this latter model is outperformed by the plain Bayesian mixtures, while in the case of large sample size it is largely equivalent.

We complete these results with the two sets

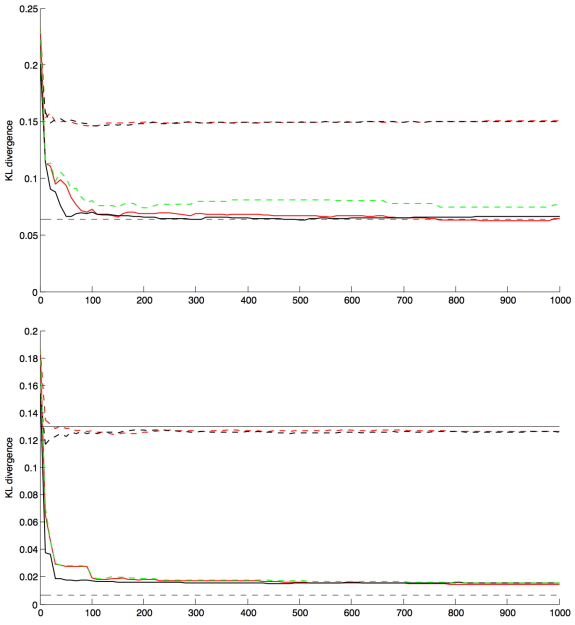

Figure 2: Example results for a target density with poly-tree structure.

of curves of Figure 2, obtained in similar conditions but when the target density factorizes according to a poly-tree structure. Overall, the previous conclusions still hold true. The main difference that we observe, is that in the case of the poly-tree target density the KL_s scores seem to converge more rapidly towards M_1 when the mixture size m increases.

4.2.2 Analysis of asymptotic behavior

Since in most trials, our learning curves stabilized around $m = 1000$, we consider interesting to provide a synthetic picture of the performances of the different methods under these "asymptotic" conditions. To this end, we show on Figure 3 overall asymptotic performances in the form of box plots of the KL_s values of the different methods.

On this figure, each box plot (box-and-whisker diagram) depicts the density of KL_s values of a particular algorithm variant (from left to right the golden standard M_1, the weighted poly-tree mixtures and the weighted tree mixtures), for a fixed sample size, but over the combination of 5 different target DAG structures, 3 different target poly-tree structures, and for each case 10 data-sets. For the sake

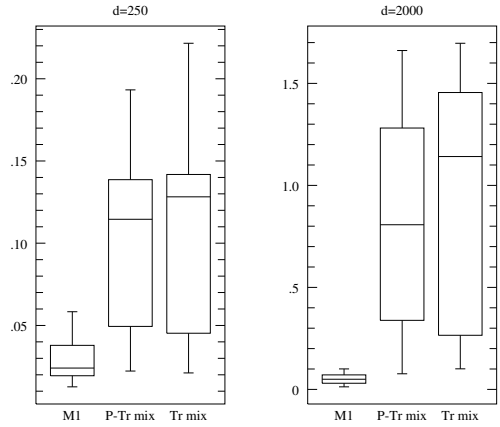

Figure 3: Synthesis of asymptotic behavior ($m = 1000$, relative KL_s wrt baseline M_0).

of interpretation, the KL_s values are normalized by dividing them by the value obtained in the same conditions (same target density, and same data-set) by the M_0 baseline method (and the latter values are not represented on the box-plots). In the left part we show the results obtained with a sample size of $d = 250$ and in the right part with a sample of size $d = 2000$.

We can synthesize these results as follows. For both large and small sample sizes the poly-tree mixtures outperform (but only very slightly) the tree mixtures; this effect is less notable in small sample conditions. In large sample conditions ($d = 2000$), the poly-tree mixtures have only a small advantage over the baseline method M_0 (their relative scores being on the average only slightly smaller than 1). However, in small sample conditions ($d = 250$), both poly-tree and tree mixtures are significantly better than the baseline, and, actually they yield KL_s scores which are already quite close to those of the the golden standard M_1.

5 Discussion

Our choice of using random mixtures of poly-trees was inspired by several considerations.

First of all, choosing the best structure in the space of poly-trees is not an adequate solution from an algorithmic point of view. Indeed, (Dasgupta, 1999) shows that finding the optimal poly-tree model is not tractable for very high dimensional spaces. On the other hand, the space of poly-trees is a priori a rather rich

space, and it is characterized by efficient inference algorithms. Hence, even a very large mixture of poly-tree densities can be queried efficiently for making inferences about the data-generating density. Furthermore, using mixtures of poly-trees allows in principle to represent any density.

In our experiments on the very simple problems with 8 binary variables, we observed however that in most cases using a mixture of poly-trees was not really better than keeping only the single best found poly-tree (except in very small sample size conditions).

Our second reason for looking at poly-tree mixtures was that we thought that these models would be more powerful than tree mixtures. Indeed, (Meila-Predoviciu, 1999) already proposed to use mixtures of tree models and has designed algorithms to find the optimal combination of tree structures and of the coefficients of the mixture during the learning phase. She jointly uses the MWST (Maximum Weight Spanning Tree) structure learning algorithm published in the late sixties (Chow and Liu, 1968) and the Expectation-Maximization algorithm for coefficients' estimation (Dempster et al., 1977). While this proposal is very elegant, we believe that it is not scalable to very large mixtures, both from the point of view of computational complexity and from the point of view of risk of over-fitting the data-set.

Our simulation results showed however that using random mixtures of poly-trees is only very marginally advantageous with respect to the use of random mixtures of trees. On the other hand, in small sample conditions the mixtures of trees or poly-trees turned out to be quite often of comparable accuracy than the golden standard $M1$, and in general largely superior to the complete structure baseline M_0.

Concerning the weighting scheme, our experiments also confirmed that uniform mixtures of randomized poly-tree or tree structured densities do not work properly in the context of density estimation. This is quite different from the observations made in the context of tree-based supervised learning, where uniform mixtures of totally randomized trees often provide very competitive results (Geurts et al., 2006). The main difference between these two contexts is that in supervised learning one can easily generate a sample of randomized trees which fit well the data-set, whereas in the context of density estimation random tree or poly-tree structures mostly strongly under-fit the data-set.

6 Summary and future works

We have proposed in this paper to transpose the "Perturb and Combine" idea celebrated in supervised learning to density estimation. We have presented a generic framework for doing this, based on random mixtures of poly-tree or tree structured Bayesian networks.

The first results obtained in the context of a simple test protocol are already interesting, 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 results with respect to the basic properties of the target distributions would be of interest. Of course, these investigations should also aim at systematically comparing all these algorithm variants both from a computational complexity and from an accuracy point of view with other mixture models proposed in the literature (Meila-Predoviciu, 1999; Lowd and Domingos, 2005), with state-of-the-art optimal structure learning algorithms (Auvray and Wehenkel, 2008), and with other approaches proposed for efficient learning (Friedman et al., 1999; Brown et al., 2004) and/or efficient inference (Jaeger, 2004) in high dimensional spaces.

Nevertheless, from these first results we are tempted to conclude that, in order to effectively transpose the Perturb and Combine idea to the context of density estimation, it will be necessary to design structure sampling algorithms which are able to efficiently focus on structures that can be fitted well enough to the available data-set. In this respect, one straightforward idea would be to transpose the Bagging idea of (Breiman, 1996) to the density estimation

context. In particular, we suggest that the use of bootstrap sampling in combination with the efficient algorithm of finding the optimal tree model, i.e. solving Equation (5) in the tree space $\mathcal{P}^1$ using the MWST algorithm, could be a very promising direction.

Another more generic direction of research, is to adapt importance sampling approaches (e.g. the cross-entropy method (Rubinstein and Kroese, 2004)) in order to generate randomized ensembles of simple structures (trees, polytrees, etc.) that fit well the given data-set.

In a later stage, we intend to extend these algorithms to the case of continuous random variables as well as when there are missing data.

Acknowledgments

This work 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.

References

V. Auvray and L. Wehenkel. 2008. Learning inclusion-optimal chordal graphs. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI-2008)*.

L. Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

L. E. Brown, I. Tsamardinos, and C. F. Aliferis. 2004. A novel algorithm for scalable and accurate bayesian network learning. *Medinfo*, 11(Pt 1):711–715.

D.M. Chickering and D. Heckerman. 1997. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212.

C.K. Chow and C.N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer.

S. Dasgupta. 1999. Learning polytrees. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 134–14, San Francisco, CA. Morgan Kaufmann.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

N. Friedman and D. Koller. 2000. Being Bayesian about network structure. In C, editor, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 201–210, SF, CA, June 30–July 3. Morgan Kaufmann Publishers.

N. Friedman, I. Nachman, and D. Pe'er. 1999. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 206–215. Morgan Kaufmann.

P. Geurts, D. Ernst, and L. Wehenkel. 2006. Extremely randomized trees. *Machine Learning*, 63(1):3–42.

J.S. Ide, F.G. Cozman, and F.T. Ramos. 2004. Generating random bayesian networks with constraints on induced width. In *ECAI*, pages 323–327.

M. Jaeger. 2004. Probabilistic decision graphs - combining verification and AI techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(Supplement-1):19–42.

S. Kullback and R. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.

D. Lowd and P. Domingos. 2005. Naive bayes models for probability estimation. In *Proceedings of the Twenty-Second International Conference (ICML 2005)*, pages 529–536. ACM.

D. Madigan and A.E. Raftery. 1994. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of The American Statistical Association*, 89:1535–1546.

D. Madigan and J. York. 1995. Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232.

M. Meila-Predoviciu. 1999. *Learning with Mixtures of Trees*. Ph.D. thesis, MIT.

J. Pearl. 1986. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288.

A. Quiroz. 1989. Fast random generation of binary, t-ary and other types of trees. *Journal of Classification*, 6(1):223–231, December. available at http://ideas.repec.org/a/spr/jclass/v6y1989i1p223-231.html.

R.W. Robinson. 1977. Counting unlabeled acyclic digraphs. In C. H. C. Little, editor, *Combinatorial Mathematics V*, volume 622 of *Lecture Notes in Mathematics*, pages 28–43, Berlin. Springer.

R.Y. Rubinstein and D.P. Kroese. 2004. *The Cross-Entropy Method. A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Information Science and Statistics. Springer.