Sub-quadratic markov tree mixture models for probability density estimation

Sourour Ammar, Philippe Leray, Louis Wehenkel

Sub-quadratic Markov tree mixture models for probability density estimation

Sourour Ammar[1], Philippe Leray[1], 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@univ-nantes.fr, philippe.leray@univ-nantes.fr
[2] Department of Electrical Engineering and Computer Science & GIGA-Research,
University of Liège, Belgium, *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 and we try to accelerate the research procedure by reducing its computation complexity below the quadratic and keepingg similar accuracy.

We empirically assess the performances of these heuristics in terms of accuracy and computation complexity, with respect to mixtures of bagged Markov trees, and single Markov tree CL built using the Chow and Liu algorithm.

Keywords: density estimation, mixture of trees, Perturb and Combine

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. 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. 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 dataset 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 dataset. 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 (e.g. Geurts et al. (2006)).

In the context of density estimation, bagging and boosting of normal distributions has been proposed by Ridgeway (2002). The link between mixture of models and bayesian modelling framework has been described in Ammar et al. (2009). In Ammar et al. (2008) 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. In Ammar et al. (2009) other comparisons were made essentially between ensembles of optimal trees derived from bootstrap copies of the dataset by the Chow and Liu algorithm (Chow and Liu (1968)), which is of quadratic complexity with respect to the number of variables (called bagging of Markov trees), and mixtures of tree structures generated in a totally randomized fashion with linear complexity in the number of variables. This work proved that *Bagged* ensembles of Markov trees significantly outperform totally randomized ensembles of Markov trees, both in terms of accuracy and speed of convergence when the number of mixture components is increased. Thus, in the present paper we focus on our methods with bagging and we study various manners to improve them by forcing the complexity of the optimization level in the *Chow Liu MWST* algorithm to come down below the quadratic and keeping the same accuracy. The main idea of this work is to weaken the Chow Liu algorithm search procedure, witch is the more expensive step, by considering only reduced ensemble of mutual information terms choosen at random. We consider two ways to randomize witch terms will be considered in the search procedure. The first one consist on choosing totally at random these terms, the second one exploate the result of previous iterations to choose part of considering terms. We assess the accuracy of these two methods empirically on a set of synthetic test problems in comparison to methods described in Ammar et al. (2009).

The rest of this paper is organized as follows. Section 2 recalls briefly the principle of learning random mixtures of models and Section 3 describes the proposed research heuristics. Section 4 collects our simulation results and Section 5 briefly concludes and highlights some directions for further research.

2 Randomized Markov tree mixtures

Randomized Markov tree mixtures was studied in Ammar et al. (2009) and Ammar et al. (2008) and applied to be an alternative to classical methods of density estimation in the context of hight-dimentional spaces and small datasets.

In the space of Markov tree structures probabilistic inference (Pearl (1986)) and parameter learning are of linear complexity in the number of variables n. Importantly, Markov tree models may be learned efficiently by the Chow and Liu algorithm which is only quadratic in the number of vertices (variables).

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

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.

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

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.

Our generic procedure for learning a random Markov tree mixture distribution from a dataset D is described by algorithm 1 (Ammar et al. (2009)). This algorithm returns the m tree-models, along with their parameters θ_{T_i} and the weights of the trees μ_i.

Algorithm 1 (Learning a Markov tree mixture)

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

Some versions of this algorithm procedures used in our experiments are further discussed in Ammar et al. (2009). We concentrate in this work on *DrawMarkovtree* procedure and we describe in the following section new heuristics based on the Perturb and Combine principle in order to reduce the complexity of previous proposed approaches.

3 *DrawMarkovtree* procedure sub-quatratic heuristics

We proposed in Ammar et al. (2009) variants for *DrawMarkovtree* procedure. The first one randomly generates unconstrained Markov trees (by sampling from a uniform density over the set of all Markov tree models).The second one builds optimal tree structures by applying the MWST (Maximum Weight Spanning Tree) structure learning algorithm (Chow and Liu (1968)) on a random bootstrap replica of the initial learning set. We demonstrated in this previous work that the consistently best method is the method which uses bagging of tree structures running with a quadratic complexity.

Thus, we propose to study this method and try to accelerate the learning procedure by keeping similar accuracy. This procedure can be decomposed in three steps : the first one consists on computing the mutual information between each pair of variables to fill an $n \times n$ symetrical mutual information matrix called MI, the second one consists on finding the maximum weight spanning tree by applying the MWST algorithm (we use the Kruskal algorithm), and finally the third consists on learning structure parameters.

The first step is quadratic on the number of variables (n^2 terms to compute) while the second step has a complexity of $E\ log(E)$ where E is the number of considered edges. If the MI matrix is totally filled ($E = n^2\ terms$), the complexity of the second step is 2 $n^2\ log(n)$. The third step is linear on the number of variables.

As we combine weak models from random trees to optimal ones learnt by applying a maximum weight spanning tree research with the MI matrix, in order to obtain a good density estimation, we propose here to apply again the Perturbe and Combine principle by using intermediate models learnt with an incomplete MI matrix. The number of terms considered K is a key parameter to estimate the total procedure complexity.

This procedure is then described by the algorithm 2.

Algorithm 2 (Naive DrawMarkovTree Subquadratic procedure)

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

$\{(i_1, i_2)\}$ represents a set of pair indices which are generated randomly according to a uniform distribution and will be used to partially fill the MI_i matrix by *ComputeMI*. *CompKruskal* takes as input the partially filled MI_i and builds the corresponding maximum weight spanning tree which will be returned by the algorithm.

We propose to consider different values of this parameter and study his impact on the accuracy of the result model. We report in this paper simulations and results for one value of the parameter $K : n\ log(n)$.

If $K = n\ log(n)$, then the first step complexity will be $n\ log(n)$. In the second step, we will consider $E = n\ log(n)$ edges to compute the corresponding maximum weight spanning tree, and then the complexity of this step will be : $E\ log(E) = n\ log(n)\ log(n\ log(n))$, which is sub-quadratic and very close to the quasi-linear.

An other idea is considered to compute sub-optimal maximum weight spanning tree by *DrawMarkovtree* procedure. This idea consists in taking advantage of the resulting Markov tree built in the previous iteration to

compute the next one. Edges indices of the Markov tree built at iteration i will be used first to fill the MI_{i+1} matrix of next iteration $i+1$, then we complete the K terms indices by generating them at random. This idea can be described by algorithm 3.

Algorithm 3 (Inertial DrawMarkovTree Subquadratic procedure)

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

We consider 2 variants of the *GenSample* function used in step 2. of algorithm 2 and 3. The first one uses the same original dataset in each iteration. The second one generates a bootstrap replica of the initial dataset.

Finally, we consider two variants for the *CompWeights* function proposed in Ammar et al. (2009), namely uniform weighting and Bayesian averaging.

4 Empirical simulations

4.1 Protocol

In order to evaluate the different heuristics proposed to ameliorate the reserach procedure complexity, 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 $n = 1000$ binary random variables. To choose a target density $\mathbb{P}_G(X)$, we first decide whether it will factorize according to a general directed acyclic graph structure. Then we use the appropriate random structure and parameter generation algorithm described in Ammar et al. (2008) to draw a structure and their parameters.

Generation of datasets For each target density and dataset size, we generated 10 different datasets by sampling values of the random variables using the Monte-Carlo method with the target structure and parameter values. We carried out simulations with dataset sizes of N = 1000 elements. Given the total number of 2^n possible configurations of our n random variables, we thus look at rather small datasets.

Learning of mixtures For a given dataset 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 $m = 150$ 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 dataset is evaluated by the approached Kullback-Leibler divergence between this density and the data-generating density $\mathbb{P}_G(X)$ used to generate the dataset.

Software implementation Our various algorithms were implemented in C++ with the Boost library (`http://www.boost.org/`) and APIs provided by the ProBT© platform (`http://bayesian-programming.org`).

Our generic algorithm can be declined by varying the tree generation function (random, algorithms 2 or 3), the learning dataset (initial data D or boostrap replica B) and the weighting coefficient (uniform or BDeu).

Table 1 summarizes the name of the different variants we will compare in the next section.

Variants name	Tree Generation	Dataset	Coefficients	Complexity
MTU	Random	D	Uniform	Linear
MTBDeu	Random	D	BDeu	Linear
FBU	Algo 2	B	Uniform	Sub-quadratic
FBBDeu	Algo 2	B	BDeu	Sub-quadratic
FDU	Algo 2	D	Uniform	Sub-quadratic
FDBDeu	Algo 2	D	Uniform	Sub-quadratic
FRBU	Algo 3	B	Uniform	Sub-quadratic
FRBBDeu	Algo 3	B	BDeu	Sub-quadratic
FRDU	Algo 3	D	Uniform	Sub-quadratic
FRDBDeu	Algo 3	D	BDeu	Sub-quadratic
CL	MWST	D	Uniform	quadratic

Table 1. Algorithms' variants name

4.2 Results

Figure 1 provides a representative set of learning curves corresponding to our simulations. 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 datasets of 1000 learning samples and seven target distributions. We thus observe in Figure 1 that our two sub-optimal tree mixture methods are clearly outperforming the random Markov tree mixture methods MTU and $MTBDeu$ in terms of accuracy when we use uniform weighting schema (FDU and FBU), but are slightly worst when we use non uniform weights.

Concerning our second proposed method, we observe from Figure 2 that all variants outperform well the random Markov tree mixture methods (which is linear in the number of variables) in terms of accuracy and are approaching the baseline CL (which is quadratic in the number of variables) when the number of mixture components grows. With all this sub-quadratic approaches, we also notice that the uniform weighting procedure is actually

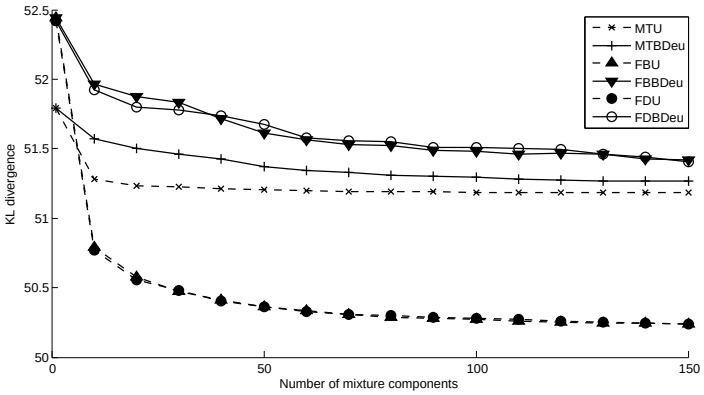

Fig. 1. Naive sub-quadratic mixtures of trees for density estimation with a DAG target distribution. 10 experiments with a sample size of 1000 for 7 random target distributions of 1000 variables. (lower is better).

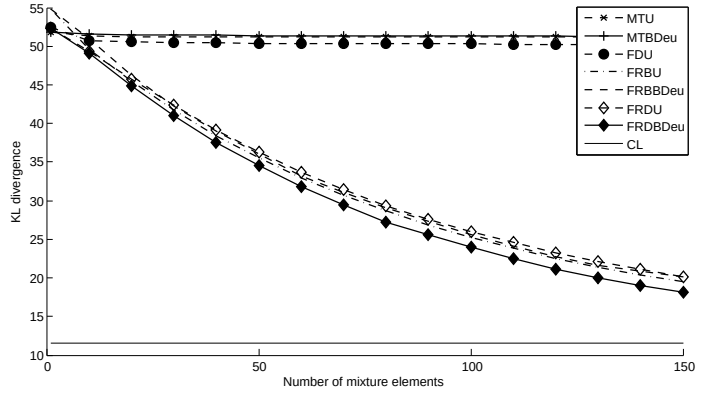

Fig. 2. Inertial sub-quadratic mixtures of trees for density estimation with a DAG target distribution. 10 experiments with a sample size of 1000 for 7 random target distributions of 1000 variables. (lower is better).

better than the one using weights based on the posterior probabilities given the dataset. Finally, we note that bagging principle do not provide better results than using the original dataset in this context of hight dimensitional problems and small datasets. All in all, the consistently best method in these trials is the method which uses uniform mixtures of sub-optimal trees built using the Chow and Liu algorithm on a partially fillet mutual information matrix whose terms are not generated at random and using the original dataset.

From a computational point of view, our proposed methods, whose complexity is $n\ log(n)\ log(n\ log(n))$ (sub-quadratic) provide better results than the linear methods and approach the single CL method whith is quadratic.

5 Summary and future works

We have proposed in this paper to apply the Perturb and Combine principle in the context of unsupervised density estimation with graphical probabilistic models by using sub-optimal models learnt with an incomplete MI matrix and the Chow and Liu algorithm. We have presented two research heuristics for doing this and provide sub-quadratic computation complexity. The perturbation was done by partially fill the MI matrix by generating at random part of this matrix terms and use it to optimize the structure component, or by bootstrapping data before filling the MI matrix.

The most interesting result is that our second proposed method with a complexity very close to the quasi-linear, provides much better results than linear randomized mixtures of Markov trees and approaches the CL method which is quadratic in the number of variables.

As future work, complexity of our methods can be further improved by using some linear approximation of spanning tree algorithm (Chazelle (2000)) in order to obtain a lower complexity. We also believe that the combination of our approaches with sequential methods such as Boosting or Markov-Chain Monte-Carlo whitch have already been applied in the context of graphical probabilistic models, might provide a very rich avenue for the design of novel density estimation algorithms.

References

AMMAR, S., LERAY, Ph., DEFOURNY, B. and WEHENKEL, L. (2008): 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)*. 9–16.

AMMAR, S., LERAY, Ph., DEFOURNY, B. and WEHENKEL, L. (2009): Probability Density Estimation by Perturbing and Combining Tree Structured Markov Networks. In: ECSQARU '09: *Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*. Springer-Verlag, 156–167.

CHAZELLE, B. (2000): A minimum spanning tree algorithm with inverse-Ackermann type complexity. *ACM 47 (6), 1028-1047.*

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

GEURTS, P., ERNST, D. and WEHENKEL, L. (2006): Extremely Randomized Trees. *Journal of Machine Learning 63 (1), 3-42.*

KULLBACK, S. and LEIBLER, R. (1951): On Information and Sufficiency. *Annals of Mathematical Statistics 22 (1), 79-86.*

PEARL, J. (1986): Fusion, Propagation, and Structuring in Belief Networks. *Artificial Intelligence 29, 241-288.*

RIDGEWAY, G. (2002): Looking for lumps: boosting and bagging for density estimation. *Journal of Computational Statistics & Data Analysis 38 (4), 379-392.*