ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

Towards sub-quadratic learning of probability density models in the form of mixtures of trees

François Schnitzler[1] and Philippe Leray[2] and Louis Wehenkel[1] *

1- Université de Liège - Department of EECS & GIGA-Research
Grande Traverse, 10 - B-4000 Liège - Belgium

2- Knowledge and Decision Team,
Laboratoire d'Informatique de Nantes Atlantique (LINA) UMR 6241,
Ecole Polytechnique de l'Université de Nantes, France

Abstract. We consider randomization schemes of the Chow-Liu algorithm from weak (bagging, of quadratic complexity) to strong ones (full random sampling, of linear complexity), for learning probability density models in the form of *mixtures of Markov trees*. Our empirical study on high-dimensional synthetic problems shows that, while bagging is the most accurate scheme on average, some of the stronger randomizations remain very competitive in terms of accuracy, specially for small sample sizes.

1 Background and motivations

A bayesian network (BN) over a finite set $\mathcal{X} = \{X_i\}_{i=1}^n$ of n discrete random variables is a graphical probabilistic model consisting of two parts [1]. The first is a directed acyclic graph over the variables (each variable corresponds to a vertex in this graph). The second is a set of conditional probability distributions (for each variable X_i, conditionally to its parents in the graph). A BN encodes a joint distribution over $\mathcal{X}$ by the product of these conditional distributions, and it may be exploited to perform probabilistic inferences over that distribution. However, exact inference as well as structure learning with BNs are NP-hard unless the skeleton of the graph is constrained [2].

Markov Trees are an interesting subclass of BNs, whose skeletons are acyclic and for which each node of the graph has (at most) one parent [1]. With Markov trees, the computational complexity of inference is linear in the number of variables, and structure learning by using the so-called *Chow-Liu* algorithm is essentially quadratic in the number of variables [3]. However, the restrictions on the structure also limit the modeling abilities of Markov trees, and, in practice, they are often not suited to represent a distribution.

A mixture model of size m consists of a set of different probability models, each of them defined over $\mathcal{X}$. To each term of the model is associated a positive weight, w_i, with the constraint $\sum_{i=1}^m w_i = 1$. A *Mixture of Trees* (MT) [4] is a particular type of mixture where each term T_i of the model is a Markov tree defined on $\mathcal{X}$. The probability $P(\mathbf{x})$ of an event $\mathbf{x}$, encoded by the MT model,

*This work was supported by a F.R.I.A. scholarship, by Wallonie-Bruxelles International, the FNRS, the French ministry of foreign and European affair, and the MESR in the framework of Hubert Curien partnerships. It 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 rests with the authors.

is equal to the weighted sum of the probabilities of this event according to each individual term of the mixture ($T_i(\mathbf{x})$): $P_{MT}(\mathbf{x}) = \sum_{i=1}^{m} w_i T_i(\mathbf{x})$.

MT models can represent a much wider class of probability densities than single trees while retaining their interesting computational properties [4], making these models attractive for scaling graphical models to high-dimensional problems. Most algorithms for learning MT models use in their inner loops the Chow-Liu algorithm [4, 5]. However, since the Chow-Liu algorithm is quadratic in the number of variables, these methods do not scale well to very high-dimensional problems, with thousands or even millions of variables, while more and more such datasets are encountered in practice. In this work we therefore investigate MT models learned by using various randomized versions of the Chow-Liu algorithm, with the aim of reducing the computational complexity, and simultaneously improving accuracy in small (i.e. realistic) sample size conditions.

The rest of this paper is organized as follows. In section 2, we describe the algorithms that we will compare, and in particular the randomized versions of the Chow-Liu algorithm. In section 3 we report our empirical investigations with these algorithms, and in section 4 we summarize and discuss further work.

2 Tree structure generation algorithms

In this section we describe algorithms using a training dataset composed of N joint observations of n random variables to derive Markov trees. We first recall the non-randomized Chow-Liu algorithm, and then describe three randomized versions of it by increasing sophistication. Notice that once a tree structure has been obtained, the associated conditional distributions are estimated from the training set (maximum a posteriori, non-informative prior); this step requires an $\mathcal{O}(nN)$ number of operations (see [5]). Notice also that in our mixtures, we always use uniform weights; indeed the experiments in [5] show that this is beneficial with respect to a bayesian weighting scheme, specially when the mixture terms are *i.i.d.* from a subset of models already well fitting the data.

2.1 Chow-Liu algorithm

This algorithm learns a Markov tree structure maximizing the likelihood of the training set [3]; it comprises three steps:

1. computation of the (symetric) matrix of empirical *Mutual Informations* (I) between all pairs of variables; this needs $\mathcal{O}(n^2 N)$ computations;
2. use of the I matrix as edge-weights to build a *Maximum Weight Spanning Tree* (MWST); this needs about $\mathcal{O}(n^2)$ operations (see for example [6]);
3. orientation, in $\mathcal{O}(n)$ operations, of the edges of the MWST by choosing an arbitrary root-node and propagating edge-directions away from it.

2.2 Random tree structure generation algorithm

This algorithm generates a tree structure totally at random, which may be done in $\mathcal{O}(n)$ operations (see [5] for a precise description of the algorithm).

ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

2.3 Random edge selection algorithm

A way to improve the complexity of the Chow-Liu algorithm is to reduce the number of (non zero) terms computed in its first step. The random edge selection algorithm does this by randomly selecting a subset of a priori fixed size of pairs of variables (edges) that will be inspected in the MWST algorithm. The complexity of this algorithm is linear in the number of edges that are drawn at random. Notice that its tree structures may be disconnected, and that their dependence on the dataset is increasing with the number of edges that are drawn.

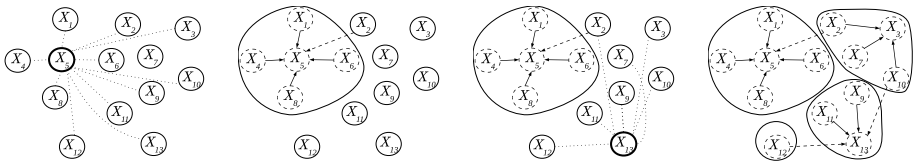
2.4 Randomized vertex clustering algorithm

In this section, we explore a less naïve idea so as to sample the potentially interesting (i.e. of large weight) edges. Our idea is based on the analogy of the MWST problem with questions such as the nearest neighbor query or the shortest path problem defined over metric spaces, were a subset of n points defines $n * (n-1)/2$ distances (or edges) between them. In metric spaces, such queries can sometimes be solved by sub-quadratic algorithms [7] exploiting the triangular inequality satisfied by distance measures: when point A is close to point B, which is far away from C, A is also likely to be far from C.

In our context, we use I to measure the "closeness" between variables, although it is not a distance measure in the mathematical sense. Still, if two pairs of variables $\{A, B\}$ and $\{A, C\}$ are close in terms of I, then B and C may as well be expected to be close in this sense. More formally, one may derive the bound: $I(B;C) \geqslant I(A;B) + I(A;C) - H(A)$, where $H(A)$ denotes the entropy of A.

Our random vertex clustering algorithm aims at avoiding the computation of the complete I matrix. To this end, it builds a clustering of the variables according to their I, and then uses subsequently only the pairs of variables that are found close to each other in this structure for the MWST algorithm. As illustrated in Fig. 1, it proceeds iteratively, until all variables are attached to a cluster. In this process, the first cluster center is chosen at random among all variables, while the subsequent ones are selected in a deterministic way as the variable among those that were not yet allocated to a cluster in the preceding iterations and that has the least I with the already existing cluster centers.

The construction of a cluster makes use of two thresholds on I: one cluster-threshold (I_C) and one neighborhood-threshold (I_N). The algorithm computes



(a) First a center (here X_5) is chosen at random and compared to all 12 other variables.

(b) Next, the 1st cluster is built. Here it has 5 members and one neighbor.

(c) The 2nd center (X_{13}, the farthest from X_5) is compared only to 7 variables.

(d) Final result, after 4 iterations. All edges considered are kept for the MWST inference.

Fig. 1: Illustration of the vertex clustering algorithm.

the mutual information I of the central cluster variable to each candidate variable (i.e. each yet unclustered variable) and identifies it as:

1. a member of the cluster, if $I > I_C$,
2. a neighbor of the cluster, if $I_C \geq I > I_N$,
3. not related to the cluster, otherwise.

Setting those thresholds can be seen as excluding potentially independent variables. The exclusion rate of independent variables can be controlled, since the empirical I for two such variables follows a χ^2 law.

In the second step of the algorithm, the Is of all potentially interesting pairs (X_i, X_j) are evaluated and used as edge-weights for the MWST algorithm: we consider as interesting those pairs that are in a same cluster or that span two neighboring clusters, two clusters being called neighbors if at least one variable of one cluster is a neighbor of the other cluster. In addition, all edges that have been evaluated during the clustering process are used as candidate edges.

The complexity of this algorithm is between linear and quadratic in the number of variables, depending on the numerical values of I_C and I_N .

3 Experimental results

The algorithms described in the previous section were applied to synthetic problems, following the methodology from [5]. The results presented here for each N were averaged on 10 datasets times 10 target distributions over 1000 variables, and we report them for mixtures of various sizes.

Method from Sec. 2.4 was applied by using for I_C (respectively I_N) the percentile 0.5 (respectively 5) of the χ^2 distribution. To assess its interest versus the random draw of edges of Sec. 2.3, this latter method was constrained to sample the same percentage (35%) of all possible edges than the former. As baselines, we used: (1) a single MWST built using the Chow-Liu algorithm; (2) mixtures of totally random tree structures; (3) mixtures obtained by applying the Chow-Liu algorithm to bootstrap copies of the dataset [5]. Indicative CPU times for training the 10×10 mixtures of size $m = 100$ are given in Table 1.

We assessed the accuracy of the studied algorithms by using a Monte-Carlo approximation of the Kullback-Leibler (KL) divergence [8] between the target distribution P_t and the learned one P_l, computed by

$$\hat{D}_{KL}(P_t||P_l) = \sum_{\mathbf{x} \square P_t} \log \frac{P_t(\mathbf{x})}{P_l(\mathbf{x})}.$$

We used two sets of 60,000 samples to check convergence of these estimations.

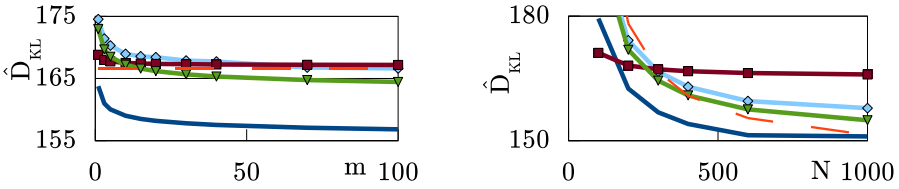
Rand. trees	Rand. edge selection	Rand. vertex clustering	Bagging
2,063 s	64,569 s	59,687 s	168,703 s

Table 1: Training CPU times, cumulated on 100 data sets of 1000 samples (MacOS X; Intel dual 2 GHz; 4GB DDR3; GCC 4.0.1)

3.1 Results

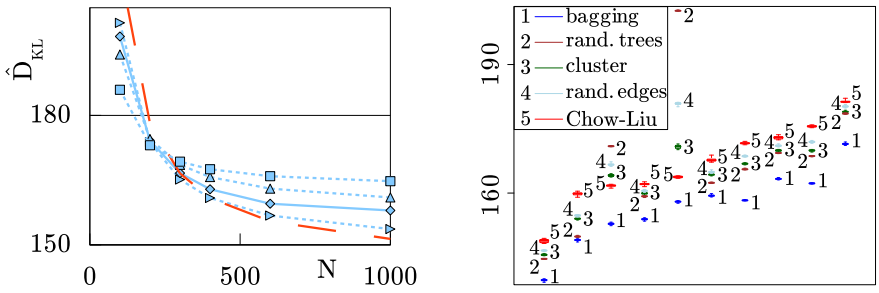
Figures 2 display the $\hat{D}_{KL}$ values of models built with all algorithms for growing mixture sizes m (2(a)), sample sizes N (2(b)) and degree of randomization. From Figure 2(a) we observe that the more sophisticated methods tend to converge slower. From Figure 2(b), we observe that bagging yields the best performances except for $N < 50$, and that edge sampling (respectively, vertex clustering) outperforms the single Chow-Liu tree for $N < 200$ (respectively, $N < 400$).

From both figures it appears clearly that the more clever detection of the problem structure by the vertex clustering method (green $\bigtriangledown$) yields in all cases a worthy improvement over the naive random edge sampling strategy (light blue $\diamondsuit$). While the improvement is small, it is significant and it comes with virtually no additional computational cost.



(a) Effect of mixture size m ($N = 300$). (b) Effect of sample size N ($m = 100$).

Fig. 2: Performances (averages over 10 target distributions and 10 datasets), for random edge sampling ($\diamondsuit$, light blue), random vertex clustering ($\bigtriangledown$, green), tree structure bagging (dark blue), random tree structures ($\square$, maroon), and a single Chow-Liu tree (dashed, red).



(a) Effect of the %age of randomly selected edges: 60, 35, 20, 5 ($\triangleright$, $\diamondsuit$, $\square$, $\square$) (m=100). (b) Variabilities of accuracies by target distribution (m=100, N=300).

Fig. 3: KL divergences estimated by Monte-Carlo (average values over 10 target distributions with 1000 variables and 10 datasets).

We believe that the behavior of the random vertex clustering algorithm to be strongly dependent on the problem structure, and we therefore leave the study of the impact of its parameters choice for future work. We expect however that its performance is more or less proportional to the number of edges computed. As a proxy, Fig. 3(a) displays the effect of a modification of the number of edges used by the random edge sampling algorithm. Without surprise, the more edges

considered the closer the curve to that of the Chow-Liu method. At low samples sizes, the fewer edges the better, while the opposite holds for larger samples.

To check the soundness of our conclusions, Figure 3(b) indicates the variability of our accuracy assessments: for the 10 target distributions (arranged horizontally) and each method we show a box plot of 20 $\hat{D}_{KL}$ values obtained with 10 learning samples of size 300 and two test samples of size 60000. We observe that the tendencies observed previously hold with almost no exception.

4 Conclusion

In this paper, we studied empirically randomization schemes to exploit the Chow-Liu algorithm for building mixtures of trees. Our study shows that the accuracy loss is in line with the gain in complexity: the two novel sampling methods are better than the less complex random structure sampling but worse than the more complex bagging. We also observe that stronger randomization is productive when the number of samples N is much smaller than the number n of variables, which is the rule in very high-dimensional problems.

A more interesting observation can be made from the comparison of our two novel sampling algorithms: vertex clustering yields indeed a significant improvement over the random edge sampling approach, without any complexity loss. While the former algorithm is not yet a convincing alternative to the state-of-the-art, further improvements can be proposed. First, its parameter space should be explored, in order find a way to tune these parameters in an adaptive way. Also, the selection of the best among the randomly generated models could be an interesting avenue for further improving this method.

Alternatives to the vertex clustering algorithm can also be proposed, e.g. like a greedy method improving a given tree by comparing neighboring variables.

References

[1] P. Judea. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[2] G.F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, March 1990.

[3] C.I. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theory*, 14:462–467, 1968.

[4] M. Meila and M.I. Jordan. Learning with mixtures of trees. *J. Mach. Learn. Res.*, 1:1–48, 2001.

[5] S. Ammar, P. Leray, B. Defourny, and L. Wehenkel. Probability density estimation by perturbing and combining tree structured markov networks. In *Proceedings of ECSQARU*, pages 156–167, 2009.

[6] B. Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *J. ACM*, 47(6):1028–1047, 2000.

[7] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA, 1998.

[8] S. Kullback and R.A. Leibler. On information and sufficiency. *ANN MATH STAT*, 22(1):79–86, 1951.