Mixture of Markov trees for Bayesian network structure learning with small datasets in high dimensional space

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@univ-nantes.fr, philippe.leray@univ-nantes.fr

Abstract. The recent explosion of high dimensionality in datasets for several domains has posed a serious challenge to existing Bayesian network structure learning algorithms. Local search methods represent a solution in such spaces but suffer with small datasets. MMHC (Max-Min Hill-Climbing) is one of these local search algorithms where a first phase aims at identifying a possible skeleton by using some statistical association measurements and a second phase performs a greedy search restricted by this skeleton. We propose to replace the first phase, imprecise when the number of data remains relatively very small, by an application of "Perturb and Combine" framework we have already studied in density estimation by using mixtures of bagged trees.

Keywords: Bayesian networks, mixture of trees, bootstrap, structure learning

1 Introduction

Bayesian networks are probabilistic graphical models that encode a joint distribution over a set of variables by a product of conditional probability distributions, one for each variable conditionally to its parents in the directed graph. These models may be learned from data and used to perform probabilistic inferences over the encoded distribution [15]. However, learning the graphical structure of such models from data is NP-hard [7]. In general, there are two main approaches for learning Bayesian network structure from data. The search-and-score approach, with algorithms such as K2 [10] or GS [8], attempts to identify the network that maximizes a given scoring function used to indicate how well the network fits the data. The second approach, constraint-based, with algorithms such as IC [16] or PC [18], attempts to estimate conditional independences between variables using statistical independence tests.

The recent explosion of high dimensionality in datasets for several domains such as the biomedical domain with hundreds or thousands of variables, has posed a serious challenge to existing Bayesian network structure learning algorithms. These algorithms are not scalable to high dimensional spaces because

of their excessive computational complexity [5]. The local search methods, hybrid between constraint-based and score-based ones, are the most appropriate solution in such spaces.

Moreover, in the context of high dimensional space, the datasets are generally very small comparatively to the space dimension. Structure learning algorithms are known to be unstable in such context : small changes in training data can cause large changes in the learned structures. So, learning a single model from small dataset will not produce a good estimation. Several works demonstrated that in such conditions, the use of the "Perturb and Combine" principle such as bagging (model averaging and bootstrap replicas) improves considerably the results. In this direction, [6] applied the bagging principle to the greedy hill-climbing algorithm with randomized restarts.

In this paper, we propose to apply the "Perturb and Combine" idea to develop a new methodology for bayesian network structure learning in the context of high dimensional space and small datasets. We propose to first learn a mixture of trees on a set of bootstrap replicas of the original dataset, and then use this mixture to guide a local search algorithm.

The rest of this paper is organized as follows. Section 2 recalls Bayesian network structure learning framework in high dimension. Section 3 presents how can we apply the Bagging principle to learn a mixture of trees from data and Section 4 describes our proposition. Section 5 presents our experimental protocol and collects our simulation results. Section 6 concludes and highlights some directions for further research.

2 Bayesian network structure learning in high dimension

2.1 Introduction

Bayesian network structure learning is NP-hard and existing algorithms are not scalable to very high dimensional spaces. Some approaches have been proposed to provide scalable algorithms, such as the Sparse Candidate algorithm [11] that constrains the search of a score-based algorithm by limiting the set of possible parents of each variable to contain at most k candidate parents. The other scalable structure learning approaches, *local search methods*, can be seen as a generalization of the sparse candidate principle. This kind of methods consists in, first, applying statistical tests to identify local structures around a target variable (e.g. a Markov Blanket (MB) or a set of candidate parents and children (CPC)), and then in using another heuristic to learn the full structure by considering the previous local results. Many heuristics have been proposed for local structure identification, IAMB [19] and MBOR [13] for Markov blanket and MMPC (Max-Min Parent Children) [20] for set of Parents and Children.

2.2 MMHC algorithm

The Max-Min Hill-Climbing (MMHC) algorithm [21] briefly described in Algorithm 1 is one of these local search structure learning algorithm. Its first phase

Algorithm 1 MMHC algorithm

Require: data D
Ensure: a DAG structure

% Restrict
for every variable $X \in V$ **do**
$PC(X) = MMPC(X, D)$
end for
% Search
Starting from an empty graph perform Greedy Hill-Climbing with operators *add-edge*, *delete-edge*, *reverse-edge*. Only try operator *add-edge*$(Y \rightarrow X)$ if $Y \in PC(X)$

return the highest scoring DAG found

consists in identifying the set of CPC for each variable by using the MMPC algorithm. The second phase consists in using a score-based algorithm (Hill-climbing) by constraining the classical *AddEdge* operator to edges discovered by MMPC in the first phase.

MMHC is scalable to high dimensional spaces with hundreds of variables and can identify a structure with higher score in less time than the Sparse Candidate algorithm.

MMPC is used in order to reconstruct a possible skeleton of the Bayesian network. This phase relies on statistical tests on training data to detect conditionally independence between variables. In the context of high dimensional space (thousands of variables) and small data sets (few hundreds samples), detecting conditionally independence between variables from data can both require an excessive computational complexity and, more damaging, return very imprecise results.

3 Mixture of bayesian networks structured trees

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

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

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

where $\mu_i \in [0, 1]$, $\sum_{i=1}^{m} \mu_i = 1$, and $\mathbb{P}_{T_i}(X)$ is the probability density over X encoded by the graphical model composed of the Markov tree structure S_i and its parameter set $\tilde{\theta}_i$:

$$\mathbb{P}_{T_i}(X) = \mathbb{P}_{S_i, \tilde{\theta}_i}(X) = \prod_{p=1}^{n} P_{\tilde{\theta}_i}(X_p | Pa_{S_i}(X_p)),$$

Algorithm 2 Markov tree Mixture learning algorithm (MtM)

Require: dataset D, mixture size m

for $i = 1, \cdots, m$ **do**
  $D_i = \text{BootstrapReplica}(D)$
  $T_i = \mathit{BuildMarkovTreeStructure}(D_i)$
  $\tilde{\theta}_i = \mathit{LearnPars}(T_i, D)$
end for
$(\mu)_{i=1}^{m} = \mathit{CompWeights}((T_i, \tilde{\theta}_i)_{i=1}^{m}, D)$
return $\left(\mu_i, T_i, \tilde{\theta}_i\right)_{i=1}^{m}$

where $Pa_{S_i}(X_p)$ is the parent variable of X_p in the tree structure S_i.

Several versions of Markov tree mixture learning algorithm described in Algorithm 2 were proposed in [2, 4] as an alternative to classical methods for density estimation in the context of high-dimensional space and small datasets : mixtures of tree structures generated in a totally randomized fashion and ensembles of optimal trees derived from bootstrap replicas of the dataset by the Chow and Liu algorithm [9] (i.e. bagging of Markov trees). In [4, 3], we also proposed three sub-quadratic heuristics to approximate the optimal tree and then to construct mixture of trees in a sub-quadratic way. Our best heuristic (Inertial search heuristic) complexity is $n \log(n) \log(n \log(n))$. These works have fruitful results for density estimation in terms of scalability and efficiency. But result of these methods, described by a mixture of several models, cannot directly identify a single model that can be graphically visualized and interpreted.

4 MtMHC algorithm

4.1 MtMHC Principle

On the one hand, scalable structure learning algorithms like MMHC can give very unstable results with small datasets. On the other hand, scalability and robustness of Markov tree mixtures for density estimation in the context of high dimensional space and small datasets make this approach attractive in such context.

For these reasons, we propose in this work to exploit the advantages of both methods with the MtMHC algorithm described in Algorithm 3. This algorithm is very similar to MMHC algorithm, but our idea is using mixtures of Markov trees in order to identify a set of candidate parents and children instead of MMPC algorithm. This new CPC identification algorithm, named MtMPC, is described in the next section.

4.2 MtMPC algorithm

Algorithm 4 describes the use of mixtures of Markov trees in order to identify a set of candidate parents and children. Given a dataset D, we first use our MTM

Algorithm 3 MtMHC algorithm

Require: dataset D, mixture size m
Ensure: a DAG structure

% Restrict
$\{PC(X)\}_{X \in V} = MtMPC(D, m)$
% Search
Starting from an empty graph perform Greedy Hill-Climbing with operators *add-edge*, *delete-edge*, *reverse-edge*. Only try operator *add-edge*$(Y \rightarrow X)$ if $Y \in PC(X)$

return the highest scoring DAG found

Algorithm 4 MtMPC algorithm

Require: dataset D, mixture size m

$\{T_i\} = MtM(D, m)$
for every variable X in V **do**
  MtMPC(X) = $\emptyset$
  for $i = 1...m$ **do**
    MtMPC(X) = MtMPC(X) $\cup$ $Ne_{T_i}(X)$
  end for
end for

return MtMPC(X)$_{X \in V}$

algorithm described in Algorithm 2 to construct a set of m Markov models. We make here the hypothesis that the union of the Markov tree models can be a good approximation of the CPC set. We then define the CPC set of a given variable A as the union of the neighbors of this variable in each tree of the mixture $Ne_{T_i}(A)$.

Using bootstrap replicas in our mixture allows to deal with small datasets. Another related solution would have been to consider more robust conditional independence tests in MMPC algorithm such as permutation tests as proposed in [17, 22].

Note that we are only working with the Markov tree skeletons without taking into account their corresponding weights μ_i. We have demonstrated in our previous work that using uniform weights in the mixture provides the best results with small datasets, so these weights are non informative for our MtMPC algorithm.

Figure 1 illustrates an example of this algorithm. An undirected skeleton G summarizing the set of all CPC is built from the ten trees of the mixture. As an illustration, the set of $CPC(A)$ appears in blue.

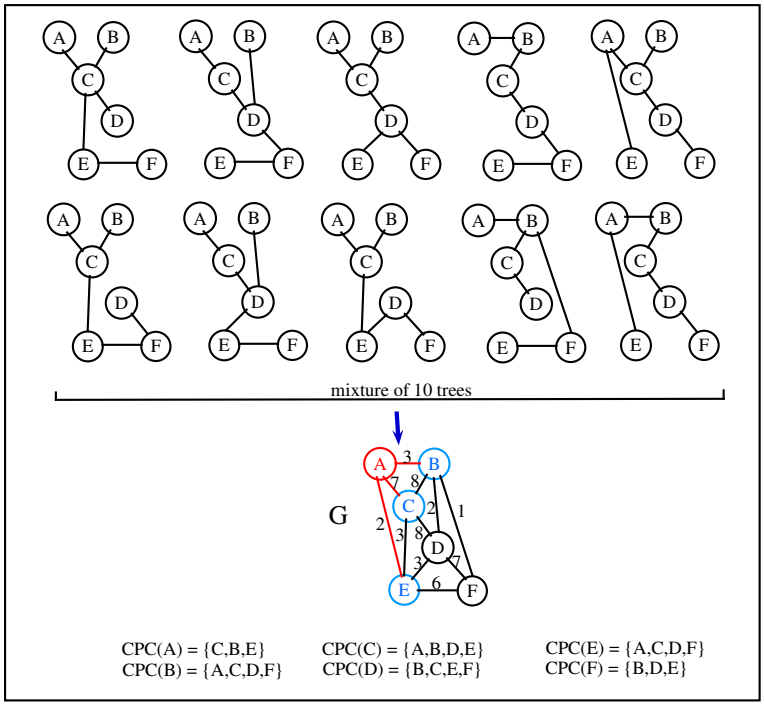

Fig. 1. Example of execution of the MtMPC algorithm

4.3 MtMHC Optimization

Because of the variability induced by boostraping data during the mixture learning, some edges only appear in a few models in the mixture. Moreover, the complexity of the second step of MtMHC algorithm (hill climbing) is directly related to the size of the PC set returned by MtMPC.

We propose one possible optimization of our MtMHC algorithm by pruning the edges non frequent in the set of Markov tree. This pruning phase is usual when we want to describe a set of graphs, as proposed in [14].

Figure 2 describes an example of CPC refinement when using this pruning optimization. When we construct the graph G from the different mixture trees, we use for each connexion in G a weight given by the number of occurence of the corresponding edge in the mixture. Then, after constructing the graph G, these weights are divised by the mixture size. So, connexions with low weight (under a given percentage) will be deleted. Figure 2 shows that connexions in G with a weight under 0.2 (threshold = 20%) ($A - E$ and $B - F$) are pruned.

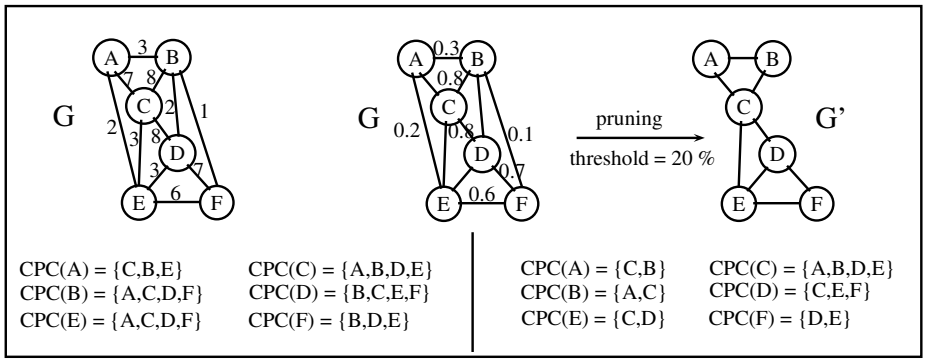

Fig. 2. Example of execution of the MtMPC algorithm with pruning phase

5 Empirical simulations and results

5.1 Experimental protocol

In order to evaluate the results of our proposition, we carried out repetitive experiments for different structures, by proceeding in the following way. All our experiments were carried out with models for a set of $n = 100$ binary random variables. To choose a target density, we first decide whether it will factorize according to a general directed acyclic graph structure. Then we use the appropriate random structure [12] and parameter generation algorithm (described in [1]) to draw a structure and their parameters.

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 $N = 50$ and 200 elements. Given the total number of 2^n possible data configurations of our n random variables, we thus look at rather small datasets in such context.

For the tree mixture learning, we tested different sizes : $m = 50$, 100, 150 and 300 trees in order to observe the potential influence of the mixture size on the quality of the result.

For this preliminary work, we concentrate our study to MtMPC results with or without pruning. Pruning threshold is set to 10% in order to illustrate the interest of the pruning optimisation.

We measure the quality of the obtained CPC sets by estimating the percentage of true positive (TP) edges with respect to the number of correct edges in the true model (edges present in the target structure and truly discovered by the algorithm) and false positive (FP) edges with respect to the number of edges absent in the true model (edges absent in the target structure and falsely discovered).

We also provide results obtained in the same conditions by MMPC algorithm using an usual statistical test (the χ^2 test with a parameter $\alpha = 5\%$), even if we

Table 1. Results obtained with MtMPC algorithm ($n = 100$, $N = 50$)

	TP	FP	TP	FP
	no pruning phase		with pruning phase	
MtMPC (m=50)	35.29 %	12.78 %	29.11 %	0.31 %
MtMPC (m=100)	39.21 %	18.58 %	28.92 %	0.27 %
MtMPC (m=150)	40.19 %	21.52 %	28.92 %	0.25 %
MtMPC (m=300)	45.09 %	23.95 %	28.82 %	0.24 %
MMPC	1.96 %	0.69 %		

Table 2. Results obtained with MtMPC algorithm ($n = 100$, $N = 200$)

	TP	FP	TP	FP
	no pruning phase		with pruning phase	
BTM-PC (m=50)	43.13 %	11.84 %	30.39 %	0.38 %
BTM-PC (m=100)	43.17 %	17.46 %	30.30 %	0.35 %
BTM-PC (m=150)	43.13 %	19.87 %	30.29 %	0.33 %
MMPC	5.88 %	1.63 %		

know that these tests are not well appropriate for small datasets. Using more sophisticated test is part of our future work.

5.2 Results

Table 1 contains MtMPC and MMPC results for a very small dataset ($N = 50$ samples) without and with pruning. Tables 2 contains similar results for small dataset ($N = 200$).

Previous work demonstrates that increasing the mixture size gives us a better estimation of the target joint distribution. This property is illustrated by the fact that the percentage of good edges (TP) discovered by our algorithm also increases in the left part of table 1 and table 2. We can also observe that it also increases the variability of the obtained trees and the number of false positives (FP).

In both tables, we can discover about 45% of the right edges in very extrem context (small datasets, $N = 50$ and 200).

The right parts of tables 1 and 2 illustrate the influence of the pruning phase. FP highly decreases to less than 0.4%, to the detriment of FP which also decreases from 45% to 30%.

As we want to plug our MtMPC results into a constraint Hill Climbing algorithm, the behavior of this pruning optimization is no so interesting. Even if we want to control the complexity of the greedy search by decreasing the number of edges of the CPC generated by MtMPC, we would like that the

pruning procedure mainly affect the FP. Pruning interesting edges is dangerous because the Hill Climbing procedure will not be able to add them again.

6 Summary and future works

We proposed in this work to apply the "Perturb and Combine" idea to develop a new methodology for bayesian network structure learning in the context of high dimensional space and small datasets. We proposed a new approach MtMHC, based on mixture of trees that have fruitful results for density estimation in such space, to guide a local search structure learning.

Our proposed algorithm for estimating the set of candidate parents and children, MtMPC, quasi-linear with respect to the number of variables, provides interesting results with very small datasets. We also proposed a potential optimization for our algorithm, but first results indicate that this optimization could be counterproductive.

As further work, we have to develop the experimentation part in several directions, by working in higher spaces, by examining the final results of MtMHC instead of the intermediate one given by MtMPC, by comparing our algorithm to specific algorithm designed for handling small datasets inspired from [17, 22].

References

1. 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). pp. 9–16 (2008)
2. Ammar, S., Leray, P., Defourny, B., Wehenkel, L.: Probability density estimation by perturbing and combining tree structured markov networks. In: Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2009). pp. 156–167. Verona, Italy (2009)
3. Ammar, S., Leray, P., Schnitzler, F., Wehenkel, L.: Sub-quadratic markov tree mixture learning based on randomizations of the chow-liu algorithm. In: the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010). pp. 17–25. Helsinki, Finland (2010)
4. Ammar, S., Leray, P., Wehenkel, L.: Sub-quadratic markov tree mixture models for probability density estimation. In: 19th International Conference on Computational Statistics (COMPSTAT 2010). pp. 673–680. Paris, France (2010)
5. Auvray, V., Wehenkel, L.: On the construction of the inclusion boundary neighbourhood for markov equivalence classes of bayesian network structures. In: Darwiche, A., Friedman, N. (eds.) Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI-02). pp. 26–35. Morgan Kaufmann Publishers, S.F., Cal. (2002)
6. Broom, B., Do, K., Subramanian, D.: Model averaging for structure learning in bayesian networks: an experimental study. Tech. rep., Computer Science Department, Rice University (2008)
7. Chickering, D., Geiger, D., Heckerman, D.: Learning bayesian networks is NP-hard. Tech. Rep. MSR-TR-94-17, Microsoft Research Technical Report (1994), http://citeseer.nj.nec.com/140425.html

8. Chickering, D.: Optimal structure identification with greedy search. Journal of Machine Learning Research 3, 507–554 (November 2003)
9. Chow, C., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory 14(3), 462–467 (1968)
10. Cooper, G., Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Machine Learning 9, 309–347 (1992)
11. Friedman, N., Nachman, I., Per, D.: Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. In: Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI). pp. 206–215 (1999)
12. Ide, J., Cozman, F., Ramos, F.: Generating random bayesian networks with constraints on induced width. In: ECAI. pp. 323–327 (2004)
13. de Morais, S.R., Aussem, A.: A novel markov boundary based feature subset selection algorithm. Neurocomputing 73(4-6), 578 – 584 (2010)
14. Nguyen, H.T., Leray, P., Ramstein, G.: Summarizing and visualizing a set of bayesian networks with quasi essential graphs. Tech. rep., University of Nantes (2011)
15. Pearl, J.: Fusion, propagation, and structuring in belief networks. Artificial Intelligence 29, 241–288 (1986)
16. Pearl, J., Verma, T.S.: A theory of inferred causation. In: Allen, J.F., Fikes, R., Sandewall, E. (eds.) KR'91: Principles of Knowledge Representation and Reasoning. pp. 441–452. Morgan Kaufmann, San Mateo, California (1991), http://citeseer.nj.nec.com/pearl91theory.html
17. Scutari, M.: Measures of Variability for Graphical Models. Ph.D. thesis, University of Padova (2011)
18. Spirtes, P., Glymour, C., Scheines, R.: Causation, prediction, and search. Springer-Verlag (1993)
19. Tsamardinos, I., Aliferis, C., Statnikov, A.: Algorithms for large scale markov blanket discovery. In: The 16th International FLAIRS Conference. pp. 376–380 (2003)
20. Tsamardinos, I., Aliferis, C., Statnikov, A.: Time and sample efficient discovery of markov blankets and direct causal relations. In: Proceedings of the 9th CAN SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 673–678 (2003)
21. Tsamardinos, I., Brown, L., Aliferis, C.: The max-min hill-climbing bayesian network structure learning algorithm. Machine Learning 65, 31–78 (2006)
22. Tsamardinos, I., Borboudakis, G.: Permutation testing improves bayesian network learning. In: Balcázar, J., Bonchi, F., Gionis, A., Sebag, M. (eds.) Machine Learning and Knowledge Discovery in Databases. Lecture Notes in Computer Science, vol. 6323, pp. 322–337. Springer Berlin, Heidelberg (2010)