Representing local structure in Bayesian networks by Boolean functions *

Yuan Zou[1], Johan Pensar[2], and Teemu Roos[1]
[1] Helsinki Institute for Information Technology HIIT, Finland
[2] Åbo Akademi University, Finland

Abstract

A number of studies on learning Bayesian networks have emphasized the importance of exploiting regularities in conditional probability distributions, i.e., local structure. In this paper, we encode local structures as linear combinations of Boolean functions. By using Lasso, we can simultaneously estimate the structure and parameters of the networks from limited data. We demonstrate that the method leads to improved performance in terms of structural correctness as well as prediction score even when the local structure in the underlying model is only implicit.

keywords: Bayesian networks; Context-specific independence; Sparsity; Logistic regression; Lasso

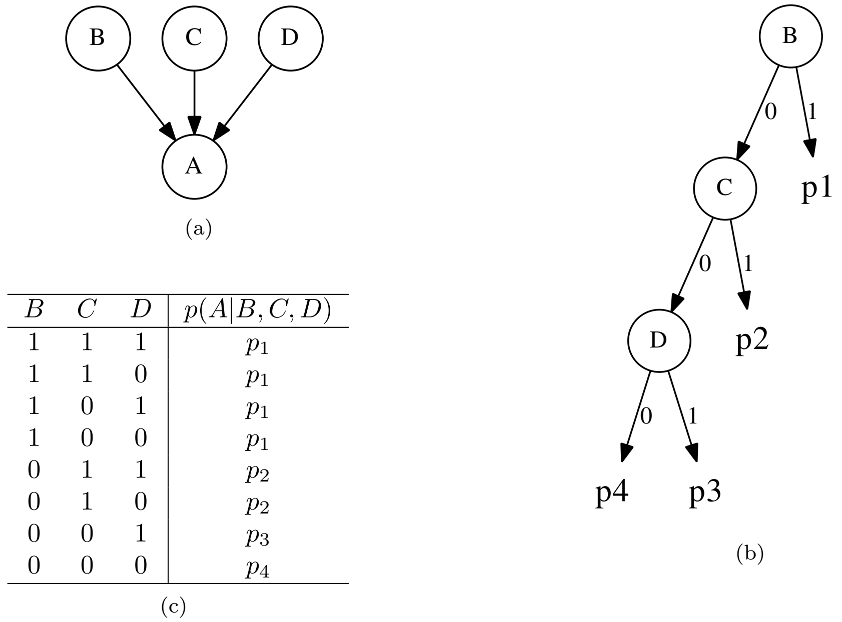
1 Introduction

A Bayesian network (BN) models conditional probability distributions among a set of random variables by a directed acyclic graph (DAG). In this paper, we only consider the case where the variables are discrete. A straightforward way of encoding the conditional probability distribution is by means of conditional probability tables (CPTs). A CPT lists the conditional distribution of a node given each combination of its parents' values. A major problem with CPTs is that the number of such combinations, which we call *parent configurations*, grows exponentially with the number of parents, which leads to a sharp increase in computational complexity and decrease in statistical precision.

On the other hand, there are cases where the conditional distributions exhibit regularities that enable their compact representation and efficient learning. To achieve more efficient representations of local structure in Bayesian networks, a number of different formulations have been proposed. In 1991, Geiger *et al.* introduced Bayesian multinets for encoding asymmetric independence in Bayesian networks (Geiger and Heckerman, 1991). Since the introduction of the concept of context-specific independence (CSI) by Boutilier et al. (1996), a set of greedy search algorithms have been developed based on this representation, see (Chickering et al., 1997; Friedman and Goldszmidt, 1998; desJardins et al., 2005). Recent work by Pensar *et al.* introduces partial conditional independence (PCI) to allow the edges of context trees to take multiple values (Pensar et al., 2016). All of the above methods assume a specific representation of local structures without considering how well it can deal with situations where the local structures are expressed in another form. Our aim is to construct a more generic representation that allows compact representations of many different types of local structures, including, for example, context trees. Finally we mention that an approach to classification of time-delayed Gaussian networks by means of transfer learning is presented in (Chaturvedi et al., 2015): transfer learning could also be used to leverage regularities in learning CPTs in Bayesian networks.

In this paper, we propose an approach to modeling local structures by using linear combinations of Boolean functions and propose a learning algorithm based on the Lasso (the least absolute shrinkage and selection operator) (Tibshirani, 1996) and an ordering-based heuristic.

[*] To be published in *Pattern Recognition Letters*.



(a)

(b)

B	C	D	$p(A\|B,C,D)$
1	1	1	p_1
1	1	0	p_1
1	0	1	p_1
1	0	0	p_1
0	1	1	p_2
0	1	0	p_2
0	0	1	p_3
0	0	0	p_4

(c)

Figure 1: An example of a Bayesian network with local structure: (a) A Bayesian network over four nodes. (b) A context tree for the conditional distributions of node A. (c) The full CPT for the context tree in (b). Note that only four distinct probability values, $p_1, \ldots, p_4$ occur in the CPT.

2 Learning local structure by Lasso

2.1 Model formulation

To capture the regularities of CPTs, we introduce a method that expresses the interactions among parents by Boolean functions. To achieve this, we define a logistic regression model as follows: for a vector of m binary variables $\vec{x} = \{x_1, x_2, \ldots, x_m\}$ and a binary response variable y, the model is

$$g(E(y)) = \beta_0 + \sum_{i=1}^{t} \beta_i L_i(\vec{x}),$$

where $E(y)$ denotes the expectation of y conditional on the regressors $\vec{x}$, $g(E(y)) = \log(E(y)/(1-E(y)))$ is the logistic link function, $L_1, \ldots, L_t$ are Boolean functions over $\vec{x}$, and $\beta_0, \ldots, \beta_t$ are the regression coefficients. By using Lasso, we can identify the set of effective interactions as those with non-zero coefficients.

In order to construct the Boolean functions used in the above logistic model, we map the original variables into a set of binary variables using a technique introduced by Roos and Yu (2009), who illustrated the robustness of this method for Markov chain models. The method enumerates all subsets of the original variables, often limited by an additional cardinality constraint, and introduces one Boolean function for each subset. In this paper, we apply AND functions, although other basis functions can be used as well: the difference in the number of terms required in different Boolean representations can be bounded, see (Roos and Yu, 2009; Zou and Roos, 2016). Multivalued (categorical) variables can also be treated in this fashion by defining the Boolean functions on all *partial configurations* (configurations involving a subset of the variables).

For example, consider two ternary variables X_1 and X_2 with values $\{0, 1, 2\}$. There are nine (full) configurations, and sixteen partial configurations: one empty configuration, three configurations for X_1, three configurations for X_2, and the nine full configurations for the pair (X_1, X_2). These define a mapping, displayed in Eq. (1) below, from the original data to sixteen binary variables, $L_1, \ldots, L_{16}$, that are used as input to the above logistic regression

$$\begin{array}{cccccccccccccccccc}
\vec{x} & & \text{AND}(\emptyset) & \text{AND}(X_1=0) & \text{AND}(X_1=1) & \text{AND}(X_1=2) & \text{AND}(X_2=0) & \text{AND}(X_2=1) & \text{AND}(X_2=2) & \text{AND}(X_1=0,X_2=0) & \text{AND}(X_1=0,X_2=1) & \text{AND}(X_1=0,X_2=2) & \text{AND}(X_1=1,X_2=0) & \text{AND}(X_1=1,X_2=1) & \text{AND}(X_1=1,X_2=2) & \text{AND}(X_1=2,X_2=0) & \text{AND}(X_1=2,X_2=1) & \text{AND}(X_1=2,X_2=2) \\
00 & \mapsto & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
01 & \mapsto & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
02 & \mapsto & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & \mapsto & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
11 & \mapsto & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
12 & \mapsto & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
20 & \mapsto & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
21 & \mapsto & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
22 & \mapsto & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array} \tag{1}$$

model.

For the Bayesian network with binary variables in Fig. 1a, the full CPT for node A (Fig. 1c) requires eight parameters. However, a closer inspection shows that some parent configurations share the same probability distribution. For example, whenever $B = 1$, $p(A|B, C, D)$ is always p_1. Representing the conditional distribution of A by the context tree (Fig. 1b) only needs four parameters.

To control the size of the design matrix, we limit the maximum cardinality of the subsets so that, for example, only two-way or three-way interactions are allowed. Furthermore, in practice it is necessary to include in the design matrix only those Boolean functions that take non-zero value on at least one observed instance in the training data.

2.2 Algorithm

To learn the parents of a variable, X_i, we construct the design matrix by mapping the set of potential parent variables, excluding X_i, into a Boolean representation as described in Sec. 2.1. The design matrix is used as the input to logistic regression with X_i as the response. By using Lasso with a suitable shrinkage parameter, the coefficients of any Boolean functions of higher-order interactions than the true ones and any Boolean functions of variables conditionally independent of X_i will be shrunk to zero if there are enough data. While cardinality constraints are necessary to avoid creating an overly large set of covariates as mentioned above, previous work suggests that the Lasso approach is computationally feasible and leads to good probability estimates from limited data Zou and Roos (2016). We select the Lasso shrinkage parameter by the Bayesian Information Criterion (BIC) criterion.

The key difficulty in learning the parent sets that define a valid DAG, compared to, for example, learning a single regression model or a time series model as in (Roos and Yu, 2009), is the acyclicity condition that implies, e.g., that two nodes cannot be parents of each other. To ensure acyclicity, we impose an ordering of the nodes and constrain the set of parents to be such that a node, X_i, can only have as parent another node, X_j, that precedes it in the ordering $X_j \prec X_i$; this is a common approach in learning Bayesian networks, see (Teyssier and Koller, 2005). The algorithm begins by initializing the ordering and then fitting the model under the ordering constraint and evaluating the log-likelihood. It is then checked whether moving any individual variable in another position in the ordering and repeating the Lasso fitting procedure with the new ordering leads to an increase in the log-likelihood. This is repeated until convergence. As for any greedy search procedure of this kind, global optimality of the solution cannot be guaranteed. A possible approach for mitigating this problem is to do multiple runs.

More specifically, the algorithm includes the following steps:

1. *Initial ordering:* For each variable in $\{X_1, X_2, \ldots, X_m\}$, find the number of related variables out of all other variables by Lasso. Order the variables in increasing order based on the number of related variables.
2. *Main loop:* Repeat the following steps until convergence or until a maximum number of iterations, T_1, is reached.
   2.1 Learn the parent set of each variable by Lasso under the ordering constraint, and evaluate the log-likelihood. If removing any edge decreases the log-likelihood by less than a given multiplicative factor, δ, remove the edge.

   2.2 Repeat the following step while keeping the undirected network structure unchanged for a predefined number of iterations, T_2:
      - Select an arbitrary parent–child pair $X_i \to X_j$. If moving X_j in a position just before X_i and changing the directions of any edges that conflict with the modified ordering increases the log-likelihood, accept the change. (These edges include the edge from X_i to X_j and all edges that point to X_j from variables between X_i and X_j.)
3. *(Optional)* For any variable, X_i, that has more parents than a predefined threshold, K, repeat the following step until X_i has at most K parents, or no further changes are available.
   - Select a parent $X_j \to X_i$. If moving X_i in a position just before X_j and changing the directions of any edges that conflict with the modified ordering decreases the log-likelihood by a factor less than δ, accept the change.

The motivation of the last step is to restrict the maximum number of the parent sets. We recommend the following values for the tuning parameters: $T_1 = 2$–3, $T_2 = 200$–250, $\delta = 0.99$, which were used in all the experiments reported below. Generally, the number of iterations, T_1 and T_2, should be set as high as possible given the computational resources. The choice of T_1 may have a significant impact on the computation time since Step 2.1 includes the estimation of the parent set of each variable and testing for edge removals. Step 2.2 is not as expensive since the undirected structure constrains the parent sets, and hence parameter T_2 can be set much larger than T_1.

The choice of δ affects the density of the resulting graph and can be thought of as a likelihood-ratio test to avoid false positives. Its effect on the log-loss in prediction tasks tends to be quite mild as long as a value in the range $0.95 \leq \delta \leq 0.99$ is used. The effect of K is similar to that of δ in restricting the density of the graph, and it can often be chosen based on domain knowledge about the possible sizes of the parent sets. To implement the Lasso steps, we use the R package `glmnet` (Friedman et al., 2010).

3 Experiments

In this section, we present three experiments involving Bayesian networks with different types of local structure. For the Lasso-based method, the maximum order of the interactions is limited to three. We remark that this is in fact fewer than the maximum number of parents per variable in all cases considered below, but we wish to demonstrate that by using linear combinations of Boolean functions, we can actually achieve good models even when the number of parents is higher than the order of the allowed interactions.

We compare the Lasso-based method to three classic methods implemented in the R package `bnlearn` (Scutari, 2010). They include: a score based method using the BIC score and hill-climbing search; a constraint based method using conditional independence tests and the interleaved incremental association algorithm (Yaramakala and Margaritis, 2005); and the pairwise method ARACNE, which is an improved version of the Chow-Liu algorithm based on pairwise mutual information (Margolin et al., 2006). We also compare our method to methods that are designed to learn local structure: CSI by Boutilier et al. (1996) and PCI by Pensar et al. (2016), as mentioned in the introduction.

We evaluate the learned models in terms of three criteria. First, we evaluate the average log-loss per sample on a hold-out test set and subtract the log-loss achieved by the underlying true model. Second, we calculate the Hamming distance between the predicted moral graph and the true moral graph (divided by the number of edges in the true moral graph). Note

Table 1: The properties of the Bayesian networks in the three experiments.

	BN I	BN II	Alarm	Insurance	Water
No. of nodes	20	20	37	27	32
Alphabet size	3	3	2–4	2–5	3–4
Avg. no. of edges	46.2	45.7	46	52	66
Avg. indegree	2.3	2.3	1.2	1.9	2.1
Min–max indegree	1–4	1–4	2–4	2–5	3–4
Total CPT size (all)*	1049.4	1021.1	509	984	10083
Total CPT size (distinct)**	316.6	253.4	288	577	5502

*) total number of possible parent configurations

**) total number of distinct values in CPTs

that the Hamming distance measures only the global structure (edge presence/absence), and ignores the local structure. In order to measure the parsimoniousness of the models, we also count the number of parameters in each estimated model (divided by the total number of distinct parameters in the conditional probability tables of the true model).

3.1 Synthetic Bayesian networks I

In the first experiment, we generate 100 random Bayesian networks with 20 variables and alphabet size three. The indegree of each node is randomly chosen in the range from one to four (subject to the acyclicity condition). Given the parent set, we create random context trees where each leaf is associated with a distinct conditional distribution of the child variable. The conditional distributions are generated by drawing numbers from the uniform distribution between 0.1 and 0.9 and normalizing them so that they sum to one. The properties of these networks are listed in the second column (BN I) of Table 1. For each Bayesian network, we generate two training sets with sample sizes 1024 and 4096, respectively, and a test set of size 8192.

The results in Figs. 2a to 2c show that under all settings, the three local structure methods outperform the three classic methods in terms of both log-loss (by a small margin) and Hamming distance (by a larger margin). In terms of the number of parameters used, the methods yield mostly comparable sized models. All three methods designed for learning local structure perform equally well for all data sets in the first experiment.

3.2 Synthetic Bayesian networks II

In the second experiment, we follow the same procedure as in the first experiment, except that the local structures of the simulated Bayesian networks are not context trees. We pick the parent configurations that share the same conditional distribution randomly from the rows of the CPT. The properties of the simulated Bayesian networks in this experiment are listed in the third column (BN II) of Table 1.

Figs. 2d to 2f illustrates the results in the second experiment. For both sample sizes, the Lasso-based method performs better than the other methods, including CSI and PCI, which are based on the assumption that the local structures are represented as context trees.

3.3 Benchmark networks

Lastly, we generate data from three commonly used benchmark Bayesian networks: *Alarm*, *Insurance* and *Water*.[1] Unlike the Bayesian networks in the previous experiments, these networks are not explicitly designed to exhibit any local structure. The number of distinct parameters in these three networks is about half of the number of possible parent configurations, while it is about one third in the first experiment and about one quarter in the second experiment, see Table 1.

[1] `http://www.bnlearn.com/bnrepository/`

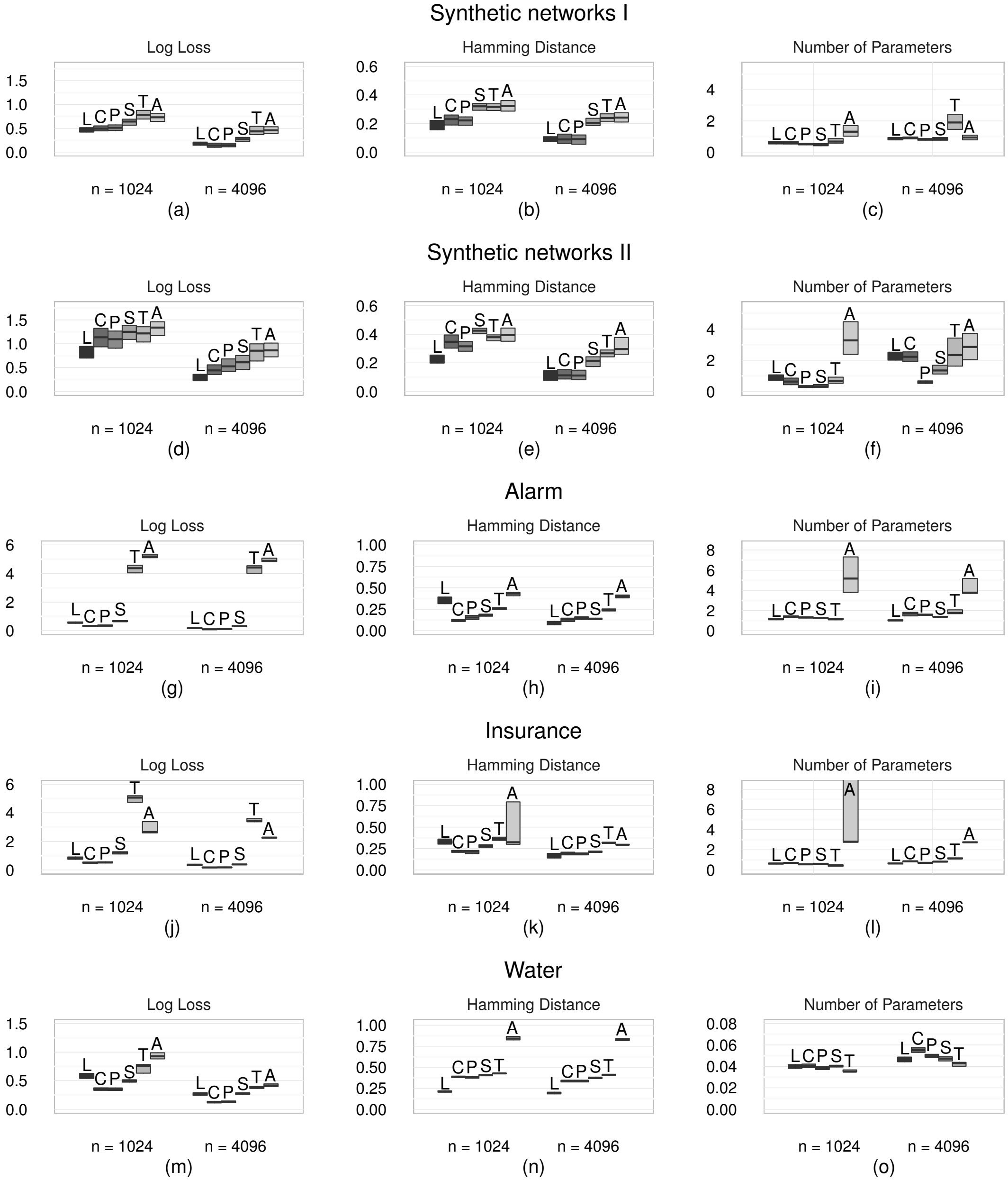


Figure 2: Box plots for comparing the performaces of the proposed Lasso-based method (L), the CSI-based method (C), the PCI-based method (P), the score based method (S), the independence test-based method (T) and ARACNE (A) in the three experiments. Large values outside the scale not shown in panels (l) and (o).

As the results shown in Figs. 2g to 2o, the results of the local structure methods, Lasso, CSI, and PCI, and the score based method, are generally similar for these three Bayesian networks. For the smaller ($n = 1024$) sample size, the Lasso method yields worse Hamming scores than the best performing methods in *Alarm* and *Insurance* but better in *Water*. For the larger sample size ($n = 4096$) its Hamming distance scores are better than those of the other methods. In terms of log-loss, for the smaller sample size, the Lasso method is slightly inferior to the other two local structure methods as well as the score-based method. For the larger sample size the differences in the log-loss of these four methods tend to be very small.

4 Conclusions

We introduced a representation of local structure in Bayesian networks using Boolean functions, and an algorithm based on Lasso and an ordering-based heuristic. The method was shown to achieve good performance in terms of structural correctness and predictive accuracy under different generating models, including benchmark networks with only implicit local structure. Broadly speaking, the results confirm the importance of considering local structure when learning Bayesian networks.

The proposed framework provides a principled way to select an appropriate model complexity, is simple to implement using existing optimization tools, and can be extended to allow various kinds of Boolean representations depending on the context. We suggest that these features make it an interesting alternative to the relatively few existing approaches.

On the other hand, it is often unclear which Boolean representation is optimal in a given context: while it is known, for instance, that some gene regulatory mechanisms can be represented as AND/OR/NOT gates and their combinations, see Shmulevich et al. (2002), it may generally be hard to choose the appropriate representation. However, as discussed in Roos and Yu (2009); Zou and Roos (2016), the number of terms required under different Boolean representations is typically of the same order of magnitude.

We plan to investigate better heuristics for dealing with structural constraints such as the acyclicity constraint considered in this work. We are also planning to further improve the scalability of the proposed method by screening techniques in order to reduce the number of potential covariates and interactions, which would enable the processing of high dimensional real-world data sets arising in, e.g., genome-wide association studies.

5 Acknowledgements

We thank Mr. Janne Leppä-aho and the anonymous reviewers for useful comments. This work was supported by the Academy of Finland (Centre-of-Excellence COIN); the Doctoral Programme in Computer Science (DoCS) and the Doctoral School in Natural Sciences at the University of Helsinki.

References

Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D., 1996. Context-specific independence in Bayesian networks, in: Horvitz, E., Jensen, F.V. (Eds.), Proc. Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI-96), Morgan Kaufmann. pp. 115–123.

Chaturvedi, I., Ong, Y.S., Arumugam, R.V., 2015. Deep transfer learning for classification of time-delayed Gaussian networks. Signal Processing 110, 250–262.

Chickering, D.M., Heckerman, D., Meek, C., 1997. A Bayesian approach to learning Bayesian networks with local structure, in: Geiger, D., Shenoy, P.P. (Eds.), Proc. Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97), Morgan Kaufmann. pp. 80–89.

desJardins, M., Rathod, P., Getoor, L., 2005. Bayesian network learning with abstraction hierarchies and context-specific independence, in: Gama, J., Camacho, R., Brazdil, P., Jorge, A., Torgo, L. (Eds.), Machine Learning: ECML 2005. Springer, pp. 485–496.

Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33, 1.

Friedman, N., Goldszmidt, M., 1998. Learning Bayesian networks with local structure, in: Jordan, M. (Ed.), Learning in Graphical Models. Springer, pp. 421–459.

Geiger, D., Heckerman, D., 1991. Advances in probabilistic reasoning, in: Bonissone, P., D'Ambrosio, B., Smets, P. (Eds.), Proc. Seventh Annual Conference on Uncertainty in Artificial Intelligence (UAI-91), Morgan Kaufmann. pp. 118–126.

Margolin, A.A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Favera, R.D., Califano, A., 2006. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics 7, S7.

Pensar, J., Nyman, H., Lintusaari, J., Corander, J., 2016. The role of local partial independence in learning of Bayesian networks. International Journal of Approximate Reasoning 69, 91–105.

Roos, T., Yu, B., 2009. Estimating sparse models from multivariate discrete data via transformed Lasso, in: Proc. Information Theory and Applications Workshop (ITA-09), IEEE. pp. 290–294.

Scutari, M., 2010. Learning Bayesian networks with the bnlearn R package. Journal of Statistical Software 35.

Shmulevich, I., Dougherty, E.R., Kim, S., Zhang, W., 2002. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. Bioinformatics 18, 261–274.

Teyssier, M., Koller, D., 2005. Ordering-based search: A simple and effective algorithm for learning Bayesian networks, in: Bacchus, F., Jaakkola, T. (Eds.), Proc. Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI-05), AUAI Press. pp. 548–549.

Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society (Series B) 58, 267–288.

Yaramakala, S., Margaritis, D., 2005. Speculative Markov blanket discovery for optimal feature selection, in: Han, J., Wah, B.W., Raghavan, V., Wu, X., Rastogi, R. (Eds.), Proc. Fifth IEEE International Conference on Data mining (ICDM-05), IEEE. pp. 809–812.

Zou, Y., Roos, T., 2016. Sparse logistic regression with logical features, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer. pp. 316–327.