BNfinder2: Bayesian networks learning and Bayesian classification

Norbert Dojer, Paweł Bednarz, Agnieszka Podsiadło i Bartek Wilczyński

September 25, 2015

Contents

1 Supplementary methods

In the present section we give a brief exposition of the algorithm implemented in BNFinder and its computational cost for two generally used scoring criteria: Minimal Description Length and Bayesian-Dirichlet equivalence. For a fuller treatment, including detailed proofs, we refer the reader to [3, 4].

1.1 Polynomial-time exact algorithm

A *Bayesian network* (BN) $\mathcal{N}$ is a representation of a joint distribution of a set of discrete random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$. The representation consists of two components:

- a directed acyclic graph $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ encoding conditional (in-)dependencies
- a family θ of conditional distributions $P(X_i|\mathbf{Pa}_i)$, where

$$\mathbf{Pa}_i = \{Y \in \mathbf{X} | (Y, X_i) \in \mathbf{E}\}$$

The joint distribution of $\mathbf{X}$ is given by

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}_i) \tag{1}$$

The problem of learning a BN is understood as follows: given a multiset of $\mathbf{X}$-instances $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ find a network graph $\mathcal{G}$ that best matches $\mathcal{D}$. The notion of a good match is formalized by means of a *scoring function* $S(\mathcal{G} : \mathcal{D})$ having positive values and minimized for the best matching network. Thus the point is to find a directed acyclic graph $\mathcal{G}$ with the set of vertices $\mathbf{X}$ minimizing $S(\mathcal{G} : \mathcal{D})$.

The BNFinder program is devoted to the case when there is no need to examine the acyclicity of the graph, for example:

- When dealing with *dynamic* Bayesian networks. A dynamic BN describes stochastic evolution of a set of random variables over discretized time. Therefore conditional distributions refer to random variables in neighboring time points. The acyclicity constraint is relaxed, because the "unrolled" graph (with a copy of each variable in each time point) is always acyclic (see [5] for more details). The following considerations apply to dynamic BNs as well.

- In case of static Bayesian Networks, the user has to supply the algorithm with a partial ordering of the vertices, restricting the set of possible edges only to the ones consistent with the ordering. BNFinder lets the user to divide the set of variables into an ordered set of disjoint subsets of variables, where edges can only exist between variables from different subsets and they have to be consistent with the ordering. If such ordering is not known beforehand, one can try to run BNFinder with different orderings and choose a network with the best overall score.

In the sequel we consider some assumptions on the form of a scoring function. The first one states that $S(\mathcal{G} : \mathcal{D})$ decomposes into a sum over the set of random variables of *local scores*, depending on the values of a variable and its parents in the graph only.

Assumption 1 (additivity) $S(\mathcal{G} : \mathcal{D}) = \sum_{i=1}^{n} s(X_i, \mathbf{Pa}_i : \mathcal{D}|_{\{X_i\} \cup \mathbf{Pa}_i})$, *where* $\mathcal{D}|_{\mathbf{Y}}$ *denotes the restriction of* $\mathcal{D}$ *to the values of the members of* $\mathbf{Y} \subseteq \mathbf{X}$.

When there is no need to examine the acyclicity of the graph, this assumption allows to compute the parents set of each variable independently. Thus the point is to find $\mathbf{Pa}_i$ minimizing $s(X_i, \mathbf{Pa}_i : \mathcal{D}|_{\{X_i\} \cup \mathbf{Pa}_i})$ for each i.

Let us fix a dataset $\mathcal{D}$ and a random variable X. We denote by $\mathbf{X}'$ the set of potential parents of X (possibly smaller than $\mathbf{X}$ due to given constraints on the structure of the network). To simplify the notation we continue to write $s(\mathbf{Pa})$ for $s(X, \mathbf{Pa} : \mathcal{D}|_{\{X\} \cup \mathbf{Pa}})$.

The following assumption expresses the fact that scoring functions decompose into 2 components: g penalizing the complexity of a network and d evaluating the possibility of explaining data by a network.

Assumption 2 (splitting) $s(\mathbf{Pa}) = g(\mathbf{Pa}) + d(\mathbf{Pa})$ *for some functions* $g, d : \mathcal{P}(\mathbf{X}) \to \mathbb{R}^+$ *satisfying* $\mathbf{Pa} \subseteq \mathbf{Pa}' \Longrightarrow g(\mathbf{Pa}) \leq g(\mathbf{Pa}')$.

This assumption is used in the following algorithm to avoid considering networks with inadequately large component g.

Algorithm 1

1. $\mathbf{Pa} := \emptyset$
2. *for each* $\mathbf{P} \subseteq \mathbf{X}'$ *chosen according to* $g(\mathbf{P})$
   (a) *if* $s(\mathbf{P}) < s(\mathbf{Pa})$ *then* $\mathbf{Pa} := \mathbf{P}$
   (b) *if* $g(\mathbf{P}) \geq s(\mathbf{Pa})$ *then return* $\mathbf{Pa}$*; stop*

In the above algorithm *choosing according to* $g(\mathbf{P})$ means choosing increasingly with respect to the value of the component g of the local score.

Theorem 1 *Suppose that the scoring function satisfies Assumptions 1-2. Then Algorithm 1 applied to each random variable finds an optimal network.*

A disadvantage of the above algorithm is that finding a proper subset $\mathbf{P} \subseteq \mathbf{X}'$ involves computing $g(\mathbf{P}')$ for all $\subseteq$-successors $\mathbf{P}'$ of previously chosen subsets. It may be avoided when a further assumption is imposed.

Assumption 3 (uniformity) $|\mathbf{Pa}| = |\mathbf{Pa}'| \Longrightarrow g(\mathbf{Pa}) = g(\mathbf{Pa}')$.

The above assumption suggests the notation $\hat{g}(|\mathbf{Pa}|) = g(\mathbf{Pa})$. The following algorithm uses the uniformity of g to reduce the number of computations of the component g.

Algorithm 2

1. $\mathbf{Pa} := \emptyset$
2. *for* $p = 1$ *to* n
   (a) *if* $\hat{g}(p) \geq s(\mathbf{Pa})$ *then return* $\mathbf{Pa}$*; stop*
   (b) $\mathbf{P} = argmin_{\{\mathbf{Y} \subseteq \mathbf{X}' : |\mathbf{Y}| = p\}} s(\mathbf{Y})$
   (c) *if* $s(\mathbf{P}) < s(\mathbf{Pa})$ *then* $\mathbf{Pa} := \mathbf{P}$

Theorem 2 *Suppose that the scoring function satisfies Assumptions 1-3. Then Algorithm 2 applied to each random variable finds an optimal network.*

1.2 Minimal Description Length and Bayesian Information Criterion

The Minimal Description Length (MDL) scoring criterion originates from information theory [7]. A network $\mathcal{N}$ is viewed here as a model of compression of a dataset $\mathcal{D}$. The optimal model minimizes the total length of the description, i.e. the sum of the description length of the model and of the compressed data.

Let us fix a dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and a random variable X. Recall the decomposition $s(\mathbf{Pa}) = g(\mathbf{Pa}) + d(\mathbf{Pa})$ of the local score for X. In the MDL score $g(\mathbf{Pa})$ stands for the length of the description of the local part of the network (i.e. the edges ingoing to X and the conditional distribution $P(X|\mathbf{Pa})$) and $d(\mathbf{Pa})$ is the length of the compressed version of X-values in $\mathcal{D}$.

Let k_Y denote the cardinality of the set $\mathcal{V}_Y$ of possible values of the random variable $Y \in \mathbf{X}$. Thus we have

$$g(\mathbf{Pa}) = |\mathbf{Pa}| \log n + \frac{\log N}{2}(k_X - 1) \prod_{Y \in \mathbf{Pa}} k_Y$$

where $\frac{\log N}{2}$ is the number of bits we use for each numeric parameter of the conditional distribution. This formula satisfies Assumption 2 but fails to satisfy Assumption 3. Therefore Algorithm 1 can be used to learn an optimal network, but Algorithm 2 cannot.

However, for many applications we may assume that all random variables have the same value set $\mathcal{V}$ of cardinality k. In this case we obtain the formula

$$g(\mathbf{Pa}) = |\mathbf{Pa}| \log n + \frac{\log N}{2}(k - 1)k^{|\mathbf{Pa}|}$$

which satisfies Assumption 3. For simplicity, we continue to work under this assumption.

Compression with respect to the network model is understood as follows: when encoding the X-values, the values of $\mathbf{Pa}$-instances are assumed to be known. Thus the optimal encoding length is given by

$$d(\mathbf{Pa}) = N \cdot H(X|\mathbf{Pa})$$

where $H(X|\mathbf{Pa}) = -\sum_{v \in \mathcal{V}} \sum_{\mathbf{v} \in \mathcal{V}^{\mathbf{Pa}}} P(v, \mathbf{v}) \log P(v|\mathbf{v})$ is the conditional entropy of X given $\mathbf{Pa}$ (the distributions are estimated from $\mathcal{D}$).

Since all the assumptions from the previous section are satisfied, Algorithm 2 may be applied to learn the optimal network. Let us turn to the analysis of its complexity.

Theorem 3 *The worst-case time complexity of Algorithm 2 for the MDL score is* $\mathcal{O}(n^{\log_k N} N \log_k N)$.

MDL is almost identical to Bayesian Information Criterion (BIC) (see [8]), which approximates Bayesian scores (see next section). The only difference is that the first element of the sum in the formula for the g component is omitted. The above theorem applies to BIC as well.

1.3 Bayesian-Dirichlet equivalence

The Bayesian-Dirichlet equivalence (BDe) scoring criterion originates from Bayesian statistics [1]. Given a dataset $\mathcal{D}$ the optimal network structure $\mathcal{G}$ maximizes the *posterior* conditional probability $P(\mathcal{G}|\mathcal{D})$. We have

$$P(\mathcal{G}|\mathcal{D}) \propto P(\mathcal{G})P(\mathcal{D}|\mathcal{G}) = P(\mathcal{G}) \int P(\mathcal{D}|\mathcal{G},\theta)P(\theta|\mathcal{G})d\theta$$

where $P(\mathcal{G})$ and $P(\theta|\mathcal{G})$ are *prior* probability distributions on graph structures and conditional distributions' parameters, respectively, and $P(\mathcal{D}|\mathcal{G},\theta)$ is evaluated due to (1).

Heckerman et al. [6], following Cooper and Herskovits [1], identified a set of independence assumptions making possible decomposition of the integral in the above formula into a product over $\mathbf{X}$. Under this condition, together with a similar one regarding decomposition of $P(\mathcal{G})$, the scoring criterion

$$S(\mathcal{G} : \mathcal{D}) = -\log P(\mathcal{G}) - \log P(\mathcal{D}|\mathcal{G})$$

obtained by taking $-\log$ of the above term satisfies Assumption 1. Moreover, the decomposition $s(\mathbf{Pa}) = g(\mathbf{Pa}) + d(\mathbf{Pa})$ of the local scores appears as well, with the components g and d derived from $-\log P(\mathcal{G})$ and $-\log P(\mathcal{D}|\mathcal{G})$, respectively.

The distribution $P((\mathbf{X},\mathbf{E})) \propto \prod_{e\in\mathbf{E}} \alpha_e$ with penalty parameters $0 < \alpha_e < 1$ is commonly used as a prior over the network structures. BNFinder sets $\alpha_{(Y,X)} = 1/k_Y$ by default. This choice results in the function

$$g(\mathbf{Pa}) = \sum_{Y\in\mathbf{Pa}} \log k_Y$$

satisfying Assumptions 2. If we moreover assume that all random variables have the same value set $\mathcal{V}$ of cardinality k, we obtain the function

$$g(\mathbf{Pa}) = |\mathbf{Pa}| \log k$$

satisfying also Assumption 3. For simplicity, we continue to work under this assumption.

However, it should be noticed that there are also used priors which satisfy neither Assumption 2 nor 3, e.g. $P(\mathcal{G}) \propto \alpha^{\Delta(\mathcal{G},\mathcal{G}_0)}$, where $\Delta(\mathcal{G},\mathcal{G}_0)$ is the cardinality of the symmetric difference between the sets of edges in $\mathcal{G}$ and in the prior network $\mathcal{G}_0$.

The *Dirichlet distribution* is generally used as a prior over the conditional distributions' parameters. It yields

$$d(\mathbf{Pa}) = \log\left(\prod_{\mathbf{v}\in\mathcal{V}^{|\mathbf{Pa}|}} \frac{\Gamma(\sum_{v\in\mathcal{V}}(H_{v,\mathbf{v}} + N_{v,\mathbf{v}}))}{\Gamma(\sum_{v\in\mathcal{V}} H_{v,\mathbf{v}})} \prod_{v\in\mathcal{V}} \frac{\Gamma(H_{v,\mathbf{v}})}{\Gamma(H_{v,\mathbf{v}} + N_{v,\mathbf{v}})}\right)$$

where Γ is the *Gamma* function, $N_{v,\mathbf{v}}$ denotes the number of samples in $\mathcal{D}$ with $X = v$ and $\mathbf{Pa} = \mathbf{v}$, and $H_{v,\mathbf{v}}$ is the corresponding *hyperparameter* of the Dirichlet distribution.

Setting all the hyperparameters to 1 yields

$$d(\mathbf{Pa}) = \log\left(\prod_{\mathbf{v}\in\mathcal{V}^{|\mathbf{Pa}|}} \frac{(k-1+\sum_{v\in\mathcal{V}} N_{v,\mathbf{v}})!}{(k-1)!} \prod_{v\in\mathcal{V}} \frac{1}{N_{v,\mathbf{v}}!}\right) =$$

$$= \sum_{\mathbf{v}\in\mathcal{V}^{|\mathbf{Pa}|}} \left(\log(k-1+\sum_{v\in\mathcal{V}} N_{v,\mathbf{v}})! - \log(k-1)! - \sum_{v\in\mathcal{V}} \log N_{v,\mathbf{v}}!\right)$$

where $k = |\mathcal{V}|$. For simplicity, we continue to work under this assumption (following Cooper and Herskovits [1]). The general case may be handled in a similar way.

The following result allows to refine the decomposition of the local score into the sum of the components g and d.

Proposition 1 *Define* $d_{min} = \sum_{v\in\mathcal{V}} (\log(k-1+N_v)! - \log(k-1)! - \log N_v!)$, *where* N_v *denotes the number of samples in* $\mathcal{D}$ *with* $X = v$. *Then* $d(\mathbf{Pa}) \geq d_{min}$ *for each* $\mathbf{Pa} \in \mathbf{X}$.

By the above proposition, the decomposition of the local score given by $s(\mathbf{Pa}) = g'(\mathbf{Pa}) + d'(\mathbf{Pa})$ with the components $g'(\mathbf{Pa}) = g(\mathbf{Pa}) + d_{min}$ and $d'(\mathbf{Pa}) = d(\mathbf{Pa}) - d_{min}$ satisfies all the assumptions required by Algorithm 2. Let us turn to the analysis of its complexity.

Theorem 4 *The worst-case time complexity of Algorithm 2 for the BDe score with the decomposition of the local score given by* $s(\mathbf{Pa}) = g'(\mathbf{Pa}) + d'(\mathbf{Pa})$ *is* $\mathcal{O}(n^{N\log_{\alpha^{-1}} k} N^2 \log_{\alpha^{-1}} k)$.

1.4 Mutual information test

The Mutual Information Test (MIT) scoring criterion originates from the concept of mutual information, belonging to the family of measures based on information theory [2]. Briefly speaking, this method combines mutual information measure and a statistical independence test based on the chi-square dustribution assosiated with it. The goodness of a fit of the particular network is computed as the total mutual information between each node and its parents. This score is then penalized by a term corresponding to the degree of statistical significance of the shared information.

Let $\mathcal{D}$ be a dataset with N observations, $\mathcal{G}$ be the dynamic bayesian network. Let $X = \{X_1, ..., X_n\}$ be the set of n variables, with each of it corresponding to $\{r_1, ..., r_n\}$ discrete states. Let's denote the set of parents of X_i in $\mathcal{G}$ with corresponding $\{r_{i1}, ..., r_{is_i}\}$ discrete states as $\mathbf{Pa}_i = \{X_{i1}, ..., X_{is_i}\}$. Then the MIT score is defined as follows [10]:

$$\mathcal{S}(\mathcal{G} : \mathcal{D}) = \sum_{i=0;\mathbf{Pa}_i\neq\emptyset}^{n} \{2N \cdot I(X_i, \mathbf{Pa}_i) - \sum_{j=1}^{s_i} \chi_{\alpha l_i \sigma_i(j)}\}$$

In this formula $I(X_i, \mathbf{Pa}_i)$ denotes the mutual information between X_i and its parents as estimated from $\mathcal{D}$ and defined as

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

$\chi_{\alpha l_i \sigma_i(j)}$ is the chi-square distribution at significance level $1-\alpha$. It is defined as the value such that

$$p(\chi^2(l_{ij}) \leq \chi_{\alpha l_i j}) = \alpha$$

The term $l_{i\sigma_i(j)}$ denotes the degrees of freedom and is defined as

$$l_{i\sigma_i(j)} = \begin{cases} (r_i - 1)(r_{i\sigma_i(j)} - 1) \prod_{k=1}^{j-1} r_{i\sigma_i(k)}, & j = 2.., s_i. \\ (r_i - 1)(r_{i\sigma_i(j)} - 1), & j = 1. \end{cases}$$

where $\sigma_i = \{\sigma_i(1), ..., \sigma_i(s_i)\}$ is any permutation of the index set $\{1...s_i\}$ of $\mathbf{Pa}_i$ such that, the number of states of the variables decreases with the increasing position in the permutation.

Recall the decomposition $S_{MIT}(\mathbf{Pa}_i) = d_{MIT}(\mathbf{Pa}_i) + g_{MIT}(\mathbf{Pa}_i)$. In this case:

$$d_{MIT}(\mathbf{Pa}_i) = 2N \cdot I(X_i, \mathbf{X}) - 2N \cdot I(X_i, \mathbf{Pa}_i)$$

$$g_{MIT}(\mathbf{Pa}_i) = \sum_{j=1}^{s_i} \chi_{\alpha l_i \sigma_i(j)}$$

Roughly, d_{MIT} measures the accuracy of representing the joint distribution of $\mathcal{D}$ by $\mathcal{G}$ while g_{MIT} measures the complexity of this representation. This decomposition satisfies Assumption 2. However, MIT score defined in this way does not satisfy Assumption 3. Therefore, we introduce an assumption that all the variables have the same number of discrete states.

Assumption 4 (uniformity) *All variables in X have the same number of discrete states k.*

Under this assumption it can be easily shown that g_{MIT} satisfies Assumption 3.

Theorem 5 *[9] The worst-case time complexity of Algorithm 2 for the MIT score under the assumption of the variables uniformity is polynomial in the number of variables.*

1.5 Continuous variables

All the scoring functions implemented in BNFinder (MDL, BIC, BDe and MIT) were originally designed for discrete variables. In order to avoid arbitrary discretization of continuous data we adapted them to deal with continuous variables

directly. Moreover, our method works also with heterogenous data sets joining together discrete and continuous variables.

The distribution of each continuous variable X is assumed to be a mixture of two normal distributions. Mixture components correspond to the two possible values (*low* and *high*) of a related hidden discrete variable X' and X is viewed as its observable reflection. Consequently, the conditional distributions of X is given by:

$$P(X|\mathbf{Pa}) = \sum_{v \in \{low, high\}} \sum_{\mathbf{v} \in \{low, high\}^{|\mathbf{Pa}|}} P(X|X' = v) P(X' = v|\mathbf{Pa}' = \mathbf{v}) P(\mathbf{Pa}' = \mathbf{v}|\mathbf{Pa})$$

Conditional distributions $P(X|X')$ are assumed to be independent for all variables X. Thus we estimate their parameters separately for each X in a preprocessing step. Estimation is based on data clustering with the *k-means* algorithm ($k = 2$, cutting the set of variable values in the median yields initial clusters). Due to the independence assumption, these parameters enable us to calculate also $P(\mathbf{Pa}'|\mathbf{Pa}) = \prod_{Y \in \mathbf{Pa}} P(Y'|Y)$. Thus the space of possible conditional distributions on continuous variables forms a family of Gaussian mixtures, parameterized by $P(X'|\mathbf{Pa}')$, conditional distributions on corresponding discrete variables.

From a technical point of view, BNFinder learns optimal network structures for these discrete variables, using scoring functions adapted to handle distributions on variable values instead of their determined values (expected values of original scores are computed). For continuous variables it gives optimal Bayesian networks from among all networks with conditional probability distributions belonging to the above defined family of Gaussian mixtures.

The following results present the complexity of our algorithm with continuous MDL and BDe scoring functions.

Theorem 6 *The worst-case time complexity of Algorithm 2 for the continuous MDL score is* $\mathcal{O}(n^{\log N} N^2)$.

Theorem 7 *The worst-case time complexity of Algorithm 2 for the continuous BDe score with the decomposition of the local score given by* $s(\mathbf{Pa}) = g'(\mathbf{Pa}) + d'(\mathbf{Pa})$ *is* $\mathcal{O}((2n)^{\frac{N}{\log \alpha^{-1}}} N)$.

1.6 Network density control

Recall that scoring functions decompose into 2 components: g penalizing the complexity of a network and d evaluating the possibility of explaining data by a network. The balance between these components influences the reliability of reconstructed relationships between variables – high g-to-d ratio results in high specificity, while low g-to-d ratio yields high sensitivity.

BNFinder has 3 mechanisms controlling this balance:

1. Option `-d` directly multiplies g-to-d ratio by a uniform factor for all pairs of variables.

2. Options `-r` and `-u` set g-to-d ratios for all edges according to specified proportion of false positive edges or of regulons having false positive regulators (thus controlling type I error rate). It is particularly useful for heterogeneous sets of potential parents (continuous and discrete, discrete with varying levels of discretization), when different types of variables require specific treatment.

3. Input dataset preamble commands `#prioredge` and `#priorvert` modify g-to-d ratios for specified network edges. They are intended to incorporate into the learning process prior knowledge regarding possible variable dependencies. This method may be combined with one of previous mechanisms.

Option `-d` modifies g-to-d ratio by virtual dataset multiplication. Remaining two mechanisms adjust components g of the scoring function. It is done through redefining the formula for g by raising parameters k_Y, the number of discretization levels of a potential parent Y to appropriate powers $w_{Y,X}$ (in the case of BDe, it is just a modification of a prior distribution over network structures). Exponents $w_{Y,X}$ are either adjusted to required type I error rate or specified in the preamble of a dataset. They must satisfy $w_{Y,X} > 0$, default values $w_{Y,X} = 1$ result in the original formula for g component.

The control of type I error rate is based on a statistical model for 1-element set of potential parents and extrapolated to all sets. In the 1-element case there are only 2 potential parent sets: $\emptyset$ and $\{Y\}$, where Y is the only potential parent of considered regulated variable X. First, BNFinder calculates the required type I error probability for edge (Y, X). When no prior distribution on the network structure is specified in the dataset preamble, all edge error probabilities equal the requested type I error rate. Otherwise they are weighted according to the inverses of prior parameters.

Under a null hypothesis H_0 that variables X and Y are independent, type I error occurs when $s(\{Y\}) < s(\emptyset)$. We define $Z_{Y,X} = d(\{Y\}) - d(\emptyset)$ and $z_{Y,X} = g(\emptyset) - g(\{Y\})$. Thus $s(\{Y\}) < s(\emptyset)$ if and only if $Z_{Y,X} < z_{Y,X}$. Note that $Z_{Y,X}$ is a function of dataset values of random variables X and Y, so it is a random variable too. On the other hand, $z_{Y,X}$ is independent of the data and monotonically depends on $w_{Y,X}$.

BNFinder randomly permutes values of Y in the dataset and calculates $Z_{Y,X}$ for each permutation. The number of permutations is chosen according to requested type I error probability and the dataset size. Moreover, it may be manually shrunk to avoid exhaustive computations. The estimate of cumulative distribution function for $Z_{Y,X}$ under H_0 assumption is derived from calculated values and $d_{min} - d(\emptyset)$, the lower bound on $Z_{Y,X}$. Based on this distribution BNFinder adjusts $w_{Y,X}$ to yield $P(Z_{Y,X} < z_{Y,X}|H_0)$ equal to the required type I error probability for edge (Y, X).

References

[1] Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[2] Luis M. de Campos. A scoring function for learning bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7:2149–2187, 2006.

[3] Norbert Dojer. Learning Bayesian Networks Does Not Have to Be NP-Hard. In Rastislav Královic and PawełUrzyczyn, editors, *Proceedings of Mathematical Foundations of Computer Science 2006*, pages 305–314. Springer-Verlag, 2006. LNCS 4162.

[4] Norbert Dojer. An efficient algorithm for learning bayesian networks from data. *Fundamenta Informaticae*, 103(1):53–67, January 2010.

[5] N Friedman, K Murphy, and S Russell. Learning the structure of dynamic probabilistic networks. In G F Cooper and S Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Inteligence*, pages 139–147, 1998.

[6] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, Sep. 1995.

[7] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10(3):269–293, 1994.

[8] Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

[9] Vinh Nguyen. The globalmit toolkit for learning optimal dynamic bayesian network user guide. 2011.

[10] Nguyen Xuan Vinh, Madhu Chetty, Ross Coppel, and Pramod P.Wangikar. Globalmit: learning globally optimal dynamic bayesian network with the mutual information test criterion. *Bioinformatics*, 27(19):2765–2766, 2011.