BIOINFORMATICS APPLICATIONS NOTE
Vol. 25 no. 2 2009, pages 286–287
doi:10.1093/bioinformatics/btn505

Systems biology

BNFinder: exact and efficient method for learning Bayesian networks

Bartek Wilczyński and Norbert Dojer*
Institute of Informatics, University of Warsaw, Poland

Received on March 4, 2008; revised on September 3, 2008; accepted on September 22, 2008
Advance Access publication September 30, 2008
Associate Editor: Thomas Lengauer

ABSTRACT

Motivation: Bayesian methods are widely used in many different areas of research. Recently, it has become a very popular tool for biological network reconstruction, due to its ability to handle noisy data. Even though there are many software packages allowing for Bayesian network reconstruction, only few of them are freely available to researchers. Moreover, they usually require at least basic programming abilities, which restricts their potential user base. Our goal was to provide software which would be freely available, efficient and usable to non-programmers.

Results: We present a BNFinder software, which allows for Bayesian network reconstruction from experimental data. It supports dynamic Bayesian networks and, if the variables are partially ordered, also static Bayesian networks. The main advantage of BNFinder is the use exact algorithm, which is at the same time very efficient (polynomial with respect to the number of observations).

Availability: The software, supplementary information and manual is available at http://bioputer.mimuw.edu.pl/software/bnf/. Besides the availability of the standalone application and the source code, we have developed a web interface to BNFinder application running on our servers. A web tutorial on different options of BNFinder is also available.

Contact: dojer@mimuw.edu.pl

1 INTRODUCTION

Computational methods of Bayesian network inference are very popular in many different areas of bioinformatics and other fields of science. Examples include: regulatory network reconstruction (Dojer *et al.*, 2006; Husmeier, 2003) where nodes represent genes and edges represent statistical dependencies which may indicate regulatory interactions; predicting gene expression from promoter sequence (Beer and Tavazoie, 2004; Segal *et al.*, 2003) where edges lead from promoter features (motif occurrences, their positions, etc.) to expression patterns (affinity to overlapping expression clusters); neural signal transduction analysis (Smith *et al.*, 2006) where network topology mimics the topology of connections between different parts of the brain and many others. Despite differences in the interpretation of network structure, the methodology of these studies is remarkably similar [see, Needham *et al.* (2007) for overview and further examples]. We aim to provide new software which could be used for different applications of Bayesian network reconstruction.

Most programs learning Bayesian networks from data are based on heuristic search techniques of identifying good models. This is due to a number of discouraging complexity results (Chickering, 1996; Chickering *et al.*, 2004; Meek, 2001) showing that, without restrictive assumptions, learning Bayesian networks from data is NP-hard with respect to the number of network vertices. On the other hand, the known exact algorithms learn the structure of optimal networks having up to 20–40 vertices (Ott *et al.*, 2004).

In an extensive comparison, Murphy (2007) lists over 50 software packages available for different applications of Bayesian networks. However, if one is searching for a free software able to infer the structure of static and dynamic Bayesian networks from data there are only two such applications:

- Banjo package (Smith *et al.*, 2006): Bayesian ANalysis with Java Objects,
- Bayes Net Toolbox (Murphy, 2002) for Matlab with an extension for dynamic Bayesian networks inference using MCMC (Husmeier, 2003).

Both of these software packages use heuristic search algorithms to find the best scoring network topology in a vast space of possible directed graphs, usually with some constraints on the maximal vertex in-degree.

2 METHODS

For a thorough treatment of the contents of the present section, we refer the reader to Supplementary Materials.

The BNFinder program is based on a novel polynomial-time algorithm for learning an optimal Bayesian network structure (Dojer, 2006). The algorithm was designed to save reasonable speed and perfect quality of learning in a wide class of problems occurring in the computational molecular biology. It works under the assumption that there is no need to examine the acyclicity of the graph, which is satisfied in the following cases:

- When dealing with dynamic Bayesian networks, a dynamic Bayesian network describes stochastic evolution of a set of random variables over discretized time. Therefore, conditional distributions refer to random variables in neighboring time points and the graph is always acyclic.
- In case of static Bayesian networks, the set of possible network structures must be restricted. BNFinder lets the user divide the set of variables into an ordered set of disjoint subsets of variables, where edges can only lead from upstream to downstream subsets. If such

*To whom correspondence should be addressed.

ordering is not known beforehand, one can try to run BNFinder with different orderings and choose a network with the best overall score.

BNFinder learns optimal networks with respect to two generally used scoring criteria: Bayesian–Dirichlet equivalence (BDe) and minimal description length (MDL). The (default) BDe score originates from Bayesian statistics and corresponds to the *posterior* probability of a network-given data. The MDL score originates from information theory and corresponds to the length of the data compressed with the compression model derived from the network structure. It also has a statistical interpretation as an approximation of the posterior probability. The algorithm works in polynomial time for both scores, but computations with the MDL are faster, especially for large datasets. However, we recommend using the BDe score due to its exactness in the statistical interpretation.

Both MDL and BDe scores were originally designed for discrete variables. Continuous variables are handled with corresponding scores, derived under the assumption that conditional distributions belong to a family of Gaussian mixtures.

BNFinder may learn either dynamic Bayesian networks (from time series data) or static ones (from independent experiment data). In the second case it is necessary to specify constraints on the network's structure forcing its acyclicity.

A special treatment is required for experiments, in which the values of some variables were perturbed (e.g. knockout experiments). Since perturbations change the structure of interactions, learning procedures have to use data selectively. BNFinder handles perturbations in the way following Dojer *et al.* (2006), i.e. for scoring sets of parents of a variable v, it takes into account only the experiments where v was not perturbed.

A prior distribution on the network structure may be specified through assigning weights to potential variable interactions in the way following Tamada *et al.* (2003). Moreover, the size of regulator sets of each variable may be bounded to a given number and the spaces of possible conditional probability distributions of selected variables may be restricted to *noisy-and* or *noisy-or* distributions.

There are important biological applications of Bayesian networks, in which usually the amount of learning data is small relative to the network size (e.g. reconstruction of gene regulatory networks from microarray data). Typically in such cases suboptimal models explain the data nearly as well as the optimal (highest scoring) one. For this reason, Friedman and Koller (2003) propose to pay attention for network *features* frequently appearing in suboptimal networks. Following this idea, BNFinder splits a potential network structure into independently learned features, each one composed of a vertex and its parent set. For each vertex BNFinder returns as an output a user-specified number of suboptimal parents features with their relative posterior probabilities. Setting this parameter to 1 causes BNFinder to learn the optimal network structure composed of the highest scoring features. Otherwise returned features constitute a class of suboptimal networks.

Output may be written in a few formats, supported in various graph and Bayesian network applications.

3 IMPLEMENTATION

The BNFinder software is implemented in the Python programming language so it can be installed and run on all popular operating systems. The only requirement is the availability of a recent version (>2.4) of the Python interpreter. Detailed installation instructions can be found on the Supplementary Web Page.

Besides of the stand-alone version of BNFinder we have made a publicly available web server which allows for using BNFinder running on our servers on users' data. The server uses a very simple web form for input and sends the results to the e-mail address provided. To save the resources, we have limited the web version to handle at most 20 variables and 500 observations.

In order to judge the performance of our software, we have compared it to the Banjo library (Smith *et al.*, 2006). As a realistic dataset, we have chosen the dataset attached as an example to the Banjo package, consisting of 20 variables and 2000 observations, published by Smith *et al.* (2006). The authors search for a dynamic Bayesian network with an in-degree of all vertices not larger than 5. It should be noted, that the number of such networks is extremely large $(((20\times19\times18\times17\times16)/(1\times2\times3\times4\times5))^{20}\sim6.4\times10^{84})$. Even though Banjo is able to analyze approximately 1 million networks per minute on a single CPU it would take it more than 10^{70} years to search through all possible networks. Thanks to the new algorithm (Dojer, 2006) our method is able to find the correct topology for the same dataset in a few hours on the same computer.

ACKNOWLEDGEMENTS

The computational resources were provided by CoE Bio-Exploratorium project: WKP 1/1.4.3/1/2004/44/44/115.

Funding: Polish Ministry of Science and Higher Education (No. PBZ-MNiI-2/1/2005 and 3 T11F 021 28, partial); Foundation for Polish Science (to B.W.).

Conflict of Interest: none declared.

REFERENCES

Beer,M.A. and Tavazoie,S. (2004) Predicting gene expression from sequence.*Cell*, **117**, 185–198.

Chickering,D.M. (1996) Learning Bayesian networks is NP-complete. In Fisher,D. and Lenz,H.-J. (eds), *Learning from Data: Artificial Inteligence and Statistics V*. Springer-Verlag.

Chickering,D.M. *et al*. (2004) Large-sample learning of Bayesian networks is NP-hard. *J. Mach. Learn. Res.*, **5**, 1287–1330.

Dojer,N. (2006) Learning Bayesian networks does not have to be NP-hard. In Královic,R. and Urzyczyn,P. (eds), *Proceedings of Mathematical Foundations of Computer Science 2006*, LNCS 4162, Springer-Verlag, pp. 305–314.

Dojer,N. *et al*. (2006) Applying dynamic Bayesian networks to perturbed gene expression data. *BMC Bioinformatics*, **7**, 249.

Friedman,N. and Koller,D. (2003) Being Bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Mach. Learn.*, **50**, 95–125.

Husmeier,D. (2003) Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, **19**, 2271–2282.

Meek,C. (2001) Finding a path is harder than finding a tree. *J. Artif. Intell. Res.*, **15**, 383–389.

Murphy,K. (2007) Software packages for graphical models – Bayesian networks. *Bull. Int. Soc. Bayesian Anal.*, **14**.

Murphy,K.P. (2002) Bayes Net Toolbox. *Technical report*. MIT Artificial Intelligence Laboratory.

Needham,C.J. *et al*. (2007) A primer on learning in Bayesian networks for computational biology. *PLoS Comput. Biol.*, **3**, e129.

Ott,S. *et al*. (2004) Finding optimal models for small gene networks. *Pac. Symp. Biocomput.*, 557–567.

Segal,E. *et al*. (2003) Genome-wide discovery of transcriptional modules from DNA sequence and gene expression. *Bioinformatics*, **19** (Suppl. 1), 273–282.

Smith,V.A. *et al*. (2006) Computational inference of neural information flow networks. *PLoS Comput. Biol.*, **2**, e161.

Tamada,Y. *et al*. (2003) Estimating gene networks from gene expression data by combining Bayesian network model with promoter element detection. *Bioinformatics*, **19** (Suppl. 2), ii227–ii236.