BNFinder2: Faster Bayesian network learning and Bayesian classification

Norbert Dojer*, Paweł Bednarz, Agnieszka Podsiadlo and Bartek Wilczyński*

Institute of Informatics, University of Warsaw, 02-097, Warsaw, Poland

ABSTRACT

Summary: Bayesian Networks (BNs) are versatile probabilistic models applicable to many different biological phenomena. In biological applications the structure of the network is usually unknown and needs to be inferred from experimental data. BNFinder is a fast software implementation of an exact algorithm for finding the optimal structure of the network given a number of experimental observations. Its second version, presented in this article, represents a major improvement over the previous version. The improvements include (i) a parallelized learning algorithm leading to an order of magnitude speed-ups in BN structure learning time; (ii) inclusion of an additional scoring function based on mutual information criteria; (iii) possibility of choosing the resulting network specificity based on statistical criteria and (iv) a new module for classification by BNs, including cross-validation scheme and classifier quality measurements with receiver operator characteristic scores.

Availability and implementation: BNFinder2 is implemented in python and freely available under the GNU general public license at the project Web site https://launchpad.net/bnfinder, together with a user’s manual, introductory tutorial and supplementary methods.

Contact: dojer@mimuw.edu.pl or bartek@mimuw.edu.pl

Supplementary information: Supplementary data are available at Bioinformatics online.

Received on December 14, 2012; revised on May 29, 2013; accepted on May 30, 2013

Bayesian Networks (BNs) are robust and versatile probabilistic models applicable to many different phenomena (Needham et al., 2007). In biology, the applications range from gene regulatory networks (Dojer et al., 2006) to protein interactions (Jansen et al., 2003) to gene expression prediction (Beer and Tavazoie, 2004) to relationships between chromatin-associated proteins (Van Steensel et al., 2010) to chromatin state prediction (Bonn et al., 2012). In many cases one needs to infer the structure of the network to build a BN model. While this problem is NP-hard in general (Chickering, 1995), it was shown by Dojer (2006) that in cases where the acyclicity of the network is ensured, it is possible to find the optimal network in polynomial time.

BNFinder (Wilczyński and Dojer, 2009) is a flexible tool for network topology learning from experimental data. Originally developed for inferring gene regulatory networks from expression data (Dojer et al., 2006), it has been since successfully applied to linking expression data with sequence motif information (Dabrowski et al., 2010), identifying histone modifications connected to enhancer activity (Bonn et al., 2012) and to predicting gene expression profiles of tissue-specific genes (Wilczyński et al., 2012). The last study is also an example of using BNFinder not as a standalone tool but as a software library. Thanks to the availability of the source code and documented API it was possible to use BNs as a part of a larger probabilistic model using Expectation-Maximization for parameter optimization.

BNFinder can be also used for classification tasks (Fig. 1). In this case the network topology is constrained to a bipartite graph between feature and class variables. The structure represents conditional dependencies of classes on selected features. This classifier model is equivalent to diagnostic BNs introduced by Kontkanen et al. (2001). The process of classification consists of several steps, carried out with dedicated BNFinder2 modules. First, to train the classifier, the optimal network structure and the conditional probability functions (CPDs) are learned with the basic bnfinder tool. Second, the bnc module makes predictions on new examples using the learned network and CPDs.

Additionally, the bnfinder-cv tool facilitates using BNFinder2 in a cross-validation framework by automatically dividing the input dataset into training and testing sets. The performance can be measured either with numerical measures such as specificity or sensitivity or by generating receiver operator characteristic (ROC) or precision-recall plots [using the ROCR package (Sing et al., 2005) or a pure python implementation, example plots shown in Fig. 1 and Supplementary Fig. S1]. All these tools, together with other BNFinder2 features, like handling mixed (both continuous and discrete) datasets, make it a complete package for easily generating classifiers for a broad range of biological datasets, as is illustrated by an application to histone modifications measurements (Bonn et al., 2012).

Although BNFinder always finds optimal networks with respect to a given score, the reliability of learned networks may vary, depending on the input data. Therefore, BNFinder attaches a couple of statistics to returned network features. This includes relative posterior probability for each set of parents and each variable as well as weighted frequency of occurrence in (sub-) optimal regulator sets for each edge.

BNFinder2 is equipped with additional quality control mechanism, allowing the user to predetermine the specificity of optimal network. Namely, the expected proportion of pairs of unrelated variables wrongly connected by an edge may be specified. Based on this proportion and the distribution of scoring function, prior distributions of network structures are adjusted.

*To whom correspondence should be addressed

© The Author 2013. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact journals.permissions@oup.com
to yield networks with the user-specified error rate (Supplementary Methods and Tutorial).

After the publication of the original BNFinder method, it was shown that the polynomial algorithm introduced by Dojer (2006) can also be applied to the Mutual Information Test (MIT), another BN scoring function based on mutual information (Vinh et al., 2011). The authors have shown that the MIT score gives more accurate results than the Minimal Description Length (MDL) score, while taking less time than Bayesian Dirichlet equivalence (BDe) score. As this compromise between accuracy and speed is desirable, we decided to adapt BNFinder to include the MIT score. This allows users to find networks with optimal MIT score not only in case of Dynamic BNs as presented by Vinh et al. (2011) but also in the case of static BNs with constrained topology. Our current implementation allows users to freely choose from all three scoring functions: MDL, BDe and MIT for static and dynamic BNs. Additionally, we provide a generalized MIT score for continuous variables (Supplementary Methods and Tutorial).

While BNFinder uses an efficient algorithm for BN structure learning, the original implementation was limited to running on a single CPU due to the limitations of the Python interpreter. Since then, multicore CPUs have become a majority and multiprocessoring support was introduced into the Python language. BNFinder2 takes advantage of these developments to facilitate using multiple CPU cores for faster computation. As the learning method used in BNFinder performs parent-set optimizations independently for each variable, it can be parallelized efficiently. Supplementary Figure S2 shows that using BNFinder2 one can achieve speed-ups almost linearly scaling with the number of cores available on different hardware platforms.

In summary, BNFinder2 represents a significant improvement over the original method in several aspects. From the user perspective, it allows for using BNFinder2 in classification setting with automated cross-validation, accuracy scoring and ROC plotting. Methodologically, it also provides a more comprehensive method for inferring networks with predefined error rate and introduces the possibility of calculating the optimal networks under the MIT score adapted to handle continuous variables as well as discrete ones. Last but not least, BNFinder2 can use parallelization on multiplecroe machines to greatly improve the running times of BN learning, especially in case of the BDe score.

**Funding:** Polish Ministry of Science and Higher Education grant [N N301 065236 to B.W. and N.D.] and Foundation for Polish Science within Homing Plus programme co-financed by the European Union—European Regional Development Fund [to A.P. and P.B.].

**Conflict of Interest:** none declared.

**REFERENCES**


