The JavaBayes system

Fabio Gagliardi Cozman
Laboratory of Automation and Systems
Av. Prof. Mello Moraes, 2231 – University of São Paulo 05508-900, São Paulo, SP, Brazil
fgcozman@usp.br http://www.cs.cmu.edu/~fgcozman/home.html

Introduction

The goal of this brief paper is to describe the JavaBayes system, a software package that manipulates graphical statistical models known as Bayesian networks (also known as Probabilistic networks, Belief networks, Causal networks). The software package is freely distributed at http://www.cs.cmu.edu/~javabayes and has been adopted in several teaching and research institutions as a vehicle for understanding, building and manipulating large statistical models with categorical variables. The web site contains more information about JavaBayes, including a user manual in HTML and postscript formats, downloadable files in ZIP and TAR formats, and a description of the statistical inference algorithm implemented in the system.

A quick tour on Bayesian networks and JavaBayes

A Bayesian network is a graphical representation for a multivariate statistical model. The idea is to use *directed acyclic graphs* as the basis for model construction. A directed graph is a collection of nodes and directed edges between nodes. If an edge goes from node X to node Y, then X is the *parent* of Y and Y is the *child* of X. The parents of X are denoted by $\mathrm{pa}(X)$, and the children or X are denoted by $\mathrm{ch}(X)$. Bayesian networks are constructed using directed graphs that are *acyclic* — graphs where it is impossible to start at a node and reach the same node following directed edges. Given a node X in a directed acyclic graph, we can easily find the *descendants* of the node (all nodes that can be reached from X).

Suppose we have a multivariate statistical model with N variables. We can represent this model by a directed acyclic graph with N nodes. We start by associating each variable with a node. We then define a conditional distribution $p(X_i|\mathrm{pa}(X_i))$ for each variable X_i (the distribution of X_i given the parents of X_i). The parents of a variable X_i are selected so that the following property is satisfied: X_i is independent of its nondescendants nonparents given its parents. The result is a graphical structure where each node "contains" a variable X_i and a distribution $p(X_i|\mathrm{pa}(X_i))$. The joint distribution is [6]:

$$p(X_1, \ldots, X_N) = \prod_{i=1}^{N} p(X_i|\mathrm{pa}(X_i))\,.$$

Such graphical models have been put to use in a variety of contexts [5, 7], most notably in the BUGS system for generic Bayesian analysis (http://www.mrc-bsu.cam.ac.uk/bugs/). Several examples that illustrate uses of Bayesian networks can be found at the JavaBayes web site.

JavaBayes is a system for creating and manipulating of Bayesian networks containing categorical variables. The system contains facilities for creating nodes, specifying variables and probability values, saving and loading networks. Networks can be loaded either from files or from the internet. The system is coded in the Java language, inheriting the portability aspects of this language and benefiting from the language facilities for operation with the internet and user interface construction. The fact that world-wide-web browsers support the Java language makes it possible to use JavaBayes directly in a web browser.

Once a Bayesian network is created or imported into the system, variables can be *observed*; that is, a variable can be fixed at a value. The collection of all observed variables and their observed values is the evidence E in the network (note that E may be empty). Two types of statistical inferences can be performed by JavaBayes:

- Computation of marginal probabilities for a variable X_q conditional on the evidence in the network. The objective is to compute $p(X_q|E)$.
- Computation of a maximum a posteriori configuration of variables X_q (where X_q can be a set of variables). The objective is to compute $\arg\max_{X_q} p(X_q|E)$.

The inferences are performed by the *variable elimination* algorithm [3, 8]. This algorithm (also known as bucket elimination) is quite similar to the peeling algorithm used in genetics for pedigree analysis [1]. The first step in any such algorithm is to discard distributions that are not needed in the computation — such distributions can be detected by efficient algorithms, thus saving considerable computational effort [4].

Note that we only need to compute $p(X_q, E)$, because the conditional $p(X_q|E)$ can be easily obtained by normalizing $p(X_q, E)$. We are then interested in computing the following expression:

$$p(X_q, E) = \sum_{\mathbf{X}_R \setminus \{\mathbf{X}_q, E\}} \left(\prod_{X_i \in \mathbf{X}_R} p(X_i|\mathrm{pa}(X_i)) \right), \tag{1}$$

where $\mathbf{X}_R$ indicates the set of variables that are effectively needed in the computation (after discarding unnecessary distributions). Now, think of the various distributions in (1) as living in a "pool" of distributions. Suppose we collect the distributions that contain variable X_1, take them off of the pool, and construct the (unnormalized but positive) function $\sum_{X_1} \left(\prod_{X_1, \mathrm{ch}(X_1)} p(X_j|\mathrm{pa}(X_j)) \right)$. Now we put this function into the distribution pool. What we have done is equivalent to "summing out" X_1; that is, we have eliminated variable X_1 from the problem. Following the same procedure, we can eliminate X_2, X_3, and so on. Eventually we obtain an unnormalized function of X_q. By normalizing this function, we obtain $p(X_q|E)$. JavaBayes uses exactly this procedure to compute posterior probability values, relying on heuristics to generate an elimination order for variables [5].

JavaBayes also contains a generalization of the original variable elimination algorithm in which intermediate results are "cached" in a tree structure, so that a sequence of inferences can be done with increased efficiency. The resulting scheme is similar to a junction tree algorithm [5]. The extended algorithm is described in a paper that can be downloaded from the JavaBayes web site [2].

The evolution of JavaBayes

The first version of JavaBayes was coded in September 1996, shortly after the XII Conference on Uncertainty in Artificial Intelligence. JavaBayes was first coded as a small library and distributed in the internet with a rudimentary user interface. The response to the first version was quite positive. Several contributions were made by users of the system; the most notable was the user interface produced by Sreekanth Nagarajan and Bruce D'Ambrosio at University of Oregon. Their graphical user interface was integrated to the inference engine and formed JavaBayes version 0.2. Eventually, the whole inference engine *and* the graphical user interface were rewritten from scratch, to become version 0.3. The basic functionality of JavaBayes has not changed since then.

The coming version 0.4 will contain a new, much improved inference engine. The new engine is substantially faster, better designed and more extensible, and it takes much less memory to run than the current engine. All the code is released with the executable, as the whole distribution is given under the Gnu-license. Consequently, everyone is welcome to contribute — you are welcome to test and to try the system, and hopefully to contribute to the evolution of JavaBayes.

The future?

JavaBayes focuses on categorical models, as these models have received great attention in artificial intelligence research. There is great room for improvement and extension. I see two areas in which JavaBayes can make a difference in an interesting way.

First, teaching Bayesian analysis is still a difficult task. There are notable teaching tools, perhaps the most interesting being the FirstBayes system developed by A. O'Hagan (http://www.shef.ac.uk/~st1ao/1b.html). JavaBayes can be an interesting tool for teaching Bayesian analysis, because Bayesian networks are easy to visualize and the system can be smoothly used through the internet. But to be a complete teaching tool, the system must be enlarged, as teaching requires the ability to display histograms, charts and other statistical niceties, and the ability to handle continuous variables.

A second contribution of JavaBayes is taking Bayesian analysis to the realm of embedded systems — devices that work inside machines, utilities or vehicles. The economic value of embedded system has been growing exponentially; predictions indicate that these systems will soon be an important part of our everyday lives. We can imagine air conditioning devices that do weather forecast, or cars that predict traffic conditions. JavaBayes is well-positioned to satisfy those needs, as it has been coded in Java (a language with some special features for embedded systems) and as it has been coded from the start to be lean and understandable.

To conclude, JavaBayes can be used now, and there are plans to keep improving it so as to support the needs of the Bayesian community and to expand the reach of Bayesian analysis.

References

[1] C. Cannings and E. A. Thompson. *Genealogical and Genetic Structure.* Cambridge University Press, Cambridge, 1981.

[2] F. G. Cozman. Generalizing variable elimination in Bayesian networks. In L. N. de Barros, R. Marcondes, F. G. Cozman, and A. H. R. Costa, eds., *Workshop on Probabilistic Reasoning in Artificial Intelligence*, pages 27-32. Editora Tec Art, São Paulo, 2000.

[3] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 75–104. MIT Press, Dordecht, The Netherlands, 1999.

[4] D. Geiger, T. Verma, and J. Pearl. Identifying independence in Bayesian networks. *Networks*, 20:507–534, 1990.

[5] F. V. Jensen. *An Introduction to Bayesian Networks.* Springer Verlag, New York, 1996.

[6] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, California, 1988.

[7] J. Whittaker. *Graphical models in applied multivariate statistics.* Wiley, 1990.

[8] N. L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, pages 301–328, 1996.

Note: Parts of this paper have been based on an invited paper sent to the Workshop on Software Support for Bayesian Analysis Systems, Colorado, 2000.