Modeling Bayesian Networks by Learning from Experts

Wim Wiegerinck

SNN, Radboud University Nijmegen, Geert Grooteplein 21,
6525 EZ Nijmegen, The Netherlands

Abstract

Bayesian network modeling by domain experts is still mainly a process of trial and error. The structure of the graph and the specification of the conditional probability tables (CPTs) are in practice often fiddled until a desired model behavior is obtained. We describe a development tool in which graph specification and CPT modeling are fully separated. Furthermore, the tuning of CPTs is handled automatically. The development tool consists of a database in which the graph description and the desired probabilistic behavior of the network are separately stored. From this database, the graph is constructed and the CPTs are numerically optimized in order to minimize the error between desired and actual behavior. The tool may be helpful in both development and maintenance of probabilistic expert systems. A demo is provided. A numerical example illustrates the methodology.

1 Introduction

Probabilistic graphical models, and in particular Bayesian networks, are nowadays well established as a modeling tool for expert systems in domains with uncertainty [1, 2]. The reason is that graphical models provide a powerful and conceptual transparent representation for probabilistic models. Their graphical structure, showing the conditional independencies between variables, allows for an easy interpretation. On the other hand, since a graphical model uniquely defines a joint probability model, the mathematical consistency and correctness are guaranteed. In other words, there are no assumptions made in the methodology. All assumptions in the model are contained in the definition of variables, the graphical structure of the model, and the parameters in the model.

The specification of a Bayesian network consists of two parts, a qualitative and a quantitative part. The qualitative part is the specification of the graphical structure of the network. The quantitative part consists of specification of the conditional probability tables (CPTs) in the network. Ideally both specifications are inferred from data. It practice, however, data is often insufficient even for the quantitative part of the specification. The alternative is then to do the specification of both parts by hand, by or in collaboration with a domain expert. In this manual specification, the determination of graph structure is often considered as a relatively straightforward task, since it usually fits well with knowledge that the domain expert has about causal relationships between variables. The quantitative part is considered a much harder or even impossible task. Often, domain experts do have ideas about at least a subset of quantitative probabilistic relations that should hold in the model. The problem is that these relations often do not directly translate in CPT parameters. An example of such a relation is a conditional probability in the 'wrong direction', from 'effect' to 'cause' (according to the graph). It is our experience that domain experts often model by fiddling the CPTs, and sometimes even both the structure and the CPTs, until some desired behavior in the network is achieved. A detailed discussion about the problem of modeling and some tools such as sensitivity analysis to guide the knowledge elicitation can be found in [3] and references therein.

In order to overcome the modeling problem methods have been proposed that automatically match model parameters to domain knowledge. One of these approaches [4] goes back to at least the early '90 's, inspired on the idea of backpropagation in neural networks. In [4], a methodology for computing derivatives of probabilities with respect to model parameters is described. These are to be used for sensitivity analysis to guide the knowledge elicitation process. As a remark, the paper also proposes to use them in gradient descent algorithm to maximize a measure of goodness-of-fit to local and global ('holistic') probability assessments. In this paper, we will further explore this direction and we will describe a general development tool that automatically generates a model from a knowledge base according to the method outlined in [4].

This paper is organized as follows. In section 2, we briefly review Bayesian networks. In section 3, we describe the method and we discuss various choices that can be made. In section 4, the tool is applied to a toy problem. We end the paper with a short discussion in section 5.

2 Probabilistic models and Bayesian networks

We restrict ourselves to probabilistic models $P(X)$ with a finite set of random variables, i.e. $X = (X_1, \ldots, X_N)$. Each variable X_i can assume a finite number of states $x_i \in \{1, \ldots, n_i\}$. Throughout the paper, we use small caps for the state of a variable, and in particular we use the notation $P(x_i) = P(X_i = x_i)$. Furthermore, we will often use sets as sub-indices to denote sub-vectors of $x = (x_1, \ldots, x_N)$ as in e.g. [5], e.g. if $\alpha = \{1, 3, 8\}$, x_α stands for $x_\alpha = (x_1, x_3, x_8)$.

In a probabilistic model, one can compute marginal distributions $P(x_a)$, and conditional distributions $P(x_a|y_b)$ by applying the standard rules of probability calculus,

$$P(x_a) = \sum_{x \setminus x_a} P(x) = \sum_{x'} \prod_{j \in a} \delta_{x'_j, x_j} P(x') \tag{1}$$

$$P(x_a|y_b) = \frac{P(x_a, y_b)}{P(y_b)} \tag{2}$$

where $\delta_{x'_j, x_j} = 1$ if $x'_j = x_j$ and 0 otherwise.

A Bayesian network is a probabilistic model P on a directed acyclic graph (DAG). Each node i in the graph corresponds to a random variable X_i together with a conditional probability table (CPT) $P(x_i|x_{\pi(i)})$, where $\pi(i)$ are the parents of i in the DAG. The joint distribution of the Bayesian network then factorizes as

$$P(x) = P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|x_{\pi(i)}) \tag{3}$$

Since a Bayesian network is a probabilistic model, marginal and conditional distributions of sets of nodes can be computed according to rules of probability calculus described above. In this paper, we assume that all required computations can be done efficiently, e.g., by using the junction tree algorithm [2].

2.1 Network parameters

A Bayesian network is specified in terms of the CPTs. Each of the CPTs in turn can be parameterized in a certain form,

$$P(x_i|x_{\pi_i}) = P_{\mathrm{PType}}(x_i|x_{\pi_i}, \vec{\theta}_i) \tag{4}$$

with PType indicating the type of parameterization, and $\vec{\theta}_i$ the parameter vector of node i with components $\theta_{i\mu}$. The dimension of the parameter vector depends on the PType and the number of parents of i. An exponential parametric form is often convenient, e.g.,

$$P_{\mathrm{Table}}(x_i|x_{\pi_i}, \vec{\theta}_i) = \frac{\exp(\theta_{i,x_i,x_{\pi(i)}})}{\sum_{x'_i} \exp(\theta_{i,x'_i,x_{\pi(i)}})} \tag{5}$$

$$P_{\mathrm{Sigm}}(x_i|x_{\pi_i}, \vec{\theta}_i) = \frac{\exp(x_i\theta_{i0} + \sum_{k\in\pi(i)} x_i\theta_{ik}x_k)}{\sum_{x'_i} \exp(x'_i\theta_{i0} + \sum_{k\in\pi(i)} x'_i\theta_{ik}x_k)} \quad (\text{for } x_i, x_k \pm 1\ ) \tag{6}$$

Other parametric CPTs, such as noisy-OR and noisy-MAX are more conveniently modeled as composition of several CPTs with additional hidden variables. For example, the noisy-OR can be parameterized by a deterministic OR applied to noisy copies of the parents [1, 2].

3 The development tool

The idea of the development tool is that the domain expert specifies all his knowledge in a database – the knowledge base. From the knowledge base, a model is then generated. Thus, the knowledge base should contain all the information that is needed for the definition of the Bayesian network. We identify several items that are relevant for the definition of a Bayesian network model.

1. Specification of the relevant variables X_i, and specification of the possible states x_i of each variable.

2. Specification of the parameterization (PType) of the CPT of each of the variables, (tables, noisy-OR, etc.). Of course this may differ from variable to variable.

3. Specification of the DAG.

4. Specification of the actual parameters of the CPTs.

The domain expert is assumed to be able to supply the information needed for item 1 to 3 in three separate tables in the knowledge base. The CPT parameterizations, i.e. the PTypes, is to be selected from a predefined library of available PTypes in the system.

The direct specification of parameters (item 4) is assumed to be too difficult for the expert. The knowledge base will contain a table with a first guess for the model parameters. These can be useful if the expert is indeed able to specify a parameter value. If the expert is certain about a parameter value, he can in addition indicate that the parameter in question is not adaptive. Otherwise, parameters may be set to a default value suggested by the tool.

The poor specification of model parameters is to be compensated by another table in the knowledge base, in which the expert can specify a number of probabilistic statements that should hold in the model. Typically such a statement is that a certain conditional probability has a certain target value, i.e., $P(X = 1|Y = 2, Z = 1) = t$ (where Y and Z do not need to be parents of X in the graph). Another type of statement is, e.g., $P(X = 1|Y = 1) < P(Z = 2|Y = 2, U = 1)$.

Given the information in the knowledge base, the procedure will be to tune the parameters such that the desired model behavior expressed in the statements is approximated as close as possible. For this purpose, an error measure between desired model behavior and actual model behavior is needed. This is achieved by expressing each statement in terms of a cost function.

3.1 Model cost

The cost functions $E_\alpha(\vec{p}^\alpha; \vec{t}^\alpha)$ for a statement α is a function of the model probabilities of interest for that statement

$$p_1^\alpha = P(x_{f(\alpha_1)}^{\alpha_1} | x_{c(\alpha_1)}^{\alpha_1}) \quad , \quad \ldots \quad , \quad p_K^\alpha = P(x_{f(\alpha_K)}^{\alpha_K} | x_{c(\alpha_K)}^{\alpha_K}) \tag{7}$$

The vector $\vec{t} = t_1, \ldots t_L$ is a set of additional parameters supplied by the domain expert, e.g. to encode the target values. The cost function is designed in such a way that in its minimum the desired probabilistic statement holds and $E = 0$.

The tool should contain a library of predefined cost functions $E_{\mathrm{EType}}(\vec{p}; \vec{t})$ where the user can choose from, e.g.,

$$E_{\mathrm{KL-1}}(p_1, t_1) = t_1 \log \frac{t_1}{p_1} + (1 - t_1) \log \frac{1 - t_1}{1 - p_1} \tag{8}$$

$$E_{\mathrm{SQ-FULL}}(\vec{p}, \vec{t}) = \sum_i (p_i - t_i)^2 \tag{9}$$

$$E_{\mathrm{INEQ}}(p_1, p_2) = \begin{cases} p_1 - p_2 & \text{if} \quad p_1 > p_2 \\ 0 & \text{if} \quad p_1 \leq p_2 \end{cases} . \tag{10}$$

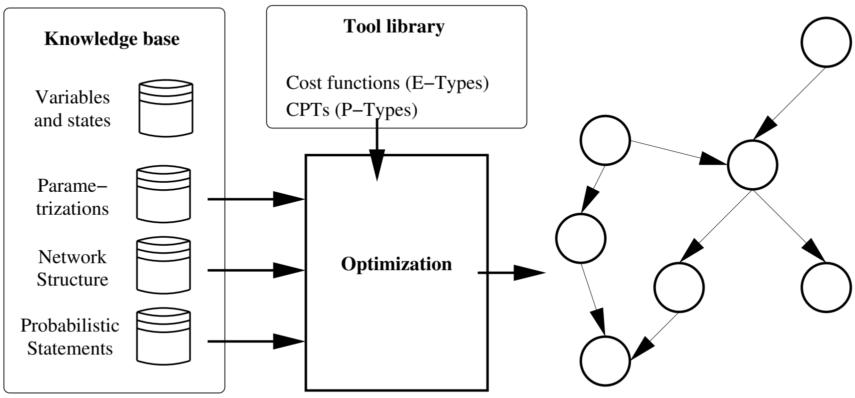

Figure 1: Development tool for Bayesian network expert system.

The function E_{INEQ} can be used to express the knowledge that a certain probability p_1 must be smaller than p_2.

The local cost functions are added to a global cost function,

$$E(\vec{\theta}) = \sum_{\alpha} w_{\alpha} E_{\mathrm{EType}(\alpha)}(\vec{p}^{\alpha}; \vec{t}^{\alpha}) \tag{11}$$

where weights $w_\alpha > 0$ are supplied by the expert to express the relative importance, or relative confidence in the statements.

Assuming that knowledge base is filled, the the parameters $\vec{\theta}$ are optimized such that the global cost function $E(\vec{\theta})$ is minimized. The optimization may be performed by a gradient based method, see appendix A. With the optimized parameters, a network can be generated, see Figure 1.

A demo system (compiled Matlab for Windows), with some example knowledge-bases can be downloaded from www.snn.ru.nl/∼wimw/bnmodeler.

4 Toy example

In this toy example, a model with 15 binary (± 1) variables is created The graphical structure is generated by linking nodes i with $i > 5$ with three parents that where randomly chosen from its predecessors. The PTypes were 'sigmoidal', as in (6). The initial parameters were set to $\theta_{i0} = -0.5$ and $\theta_{ik} = 0.5$. The model is optimized to reproduce probabilistic statements about the reversed probabilities

$$P(X_k = 1|X_i = 1) = 0.8 \ \ \forall k \in \pi(i) \tag{12}$$

$$P(X_k = 1|X_i = -1) = 0.3 \ \ \forall k \in \pi(i) \tag{13}$$

The cost function is taken to be $E_{\mathrm{KL}-1}$. The Matlab optimization took about half an hour. The statements are reproduced with a precision of about 2%. The network is stored in BayesBuilder[1] '.bbnet' format, and can be downloaded from

[1]Freely available for academic purposes from www.snn.ru.nl/nijmegen

www.snn.ru.nl/~wimw/bnmodeler/randmodel.bbnet.

5 Discussion

We described a tool for developing Bayesian networks based on a proposal by [4]. Advantages of modeling with the tool are the following: (1) it shortcuts trial and error behavior of the modeler, and therefore it facilitates model development. (2) Maintenance of the model is easier, since, for instance with new domain knowledge only records in the database that are related to this new knowledge need to be changed. By compilation, the expert system will be automatically adapted accordingly. Another possibility is to compile networks from part of the database. (3) It allows to test other different paradigms by applying different model structures without changing the probability knowledge databases.

The development tool itself is very general and flexible. The libraries with PTypes and ETypes are easily extended; data can be easily incorporated by adding the data-likelihood to the cost function; Statements about constraints e.g. with E_{INEQ}, can be included via cooling schedules (i.e. gradually increasing w_α during optimization when statement α is indicated to be a constraint).

The tool should be used with a little bit of care due to the issue of *model identifiability*. If there are far more parameters than probabilistic statements, the resulting model will depend strongly on the initial guessed parameters. Another point of care is the possibility of local minima that might obstruct the optimization.

A demo of the tool is available via the web.

A Computing the gradient for optimization

A general method to minimize the error function is by a gradient based method, such as the conjugate gradients algorithm for nonlinear optimization [6]. An important ingredient in these algorithms is the computation of the gradient of the cost function. In this section, we explain how this computation can be performed in the tool.

A.1 Gradient of the cost functions

To compute the gradient of the full E, we have to compute the partial derivatives to all the parameters $\theta_{i\mu}$,

$$\begin{aligned}\frac{\partial E(\vec{\theta})}{\partial \theta_{i\mu}} &= \sum_\alpha w_\alpha \frac{\partial E_{\mathrm{EType}(\alpha)}(\vec{p}^\alpha, \vec{t}^\alpha)}{\partial \theta_{i\mu}} \\ &= \sum_\alpha w_\alpha \sum_k \left.\frac{\partial E_{\mathrm{EType}(\alpha)}(\vec{p}, \vec{t}^\alpha)}{\partial p_k}\right|_{\vec{p}\,=\,\vec{p}^\alpha} \frac{\partial p_k^\alpha}{\partial \theta_{i\mu}} \qquad (14)\end{aligned}$$

in which $\vec{p}^\alpha$ is as in (7). Note that the functional form of the gradient of E_α with respect to $\vec{p}$ is independent of the actual value of the $\vec{p}^\alpha$. In other words, it is a property of the EType cost function,

$$G^k_{\mathrm{EType}}(\vec{p},\vec{t}) \equiv \frac{\partial E_{\mathrm{EType}}(\vec{p},\vec{t})}{\partial p_k} \tag{15}$$

Each EType gradient can simply be stored together with the Etype cost function in the library of cost functions supplied by the tool. During optimization, it can be loaded and evaluated at $(\vec{p}^\alpha, \vec{t}^\alpha)$,

$$\frac{\partial E(\vec{\theta})}{\partial \theta_{i\mu}} = \sum_\alpha w_\alpha \sum_k G^k_{\mathrm{EType}(\alpha)}(\vec{p}^\alpha, \vec{t}^\alpha) \frac{\partial p^\alpha_k}{\partial \theta_{i\mu}} \tag{16}$$

A.2 Gradient of probabilities

To proceed, we need to evaluate the partial derivatives

$$\frac{\partial p^\alpha_k}{\partial \theta_{i\mu}} = \frac{\partial P(x^{\alpha_k}_{f(\alpha_k)} | x^{\alpha_k}_{c(\alpha_k)}, \vec{\theta})}{\partial \theta_{i\mu}} \tag{17}$$

Due to the graphical structure of the DAG, only for a subset of α_k's the conditional probabilities will be depend on the value of $\vec{\theta}_i$. These relevant sets for i can be computed in advance by graphical considerations only, using the notion of *d-separation* [1]. The derivative of the conditional distribution of the relevant α_k's (while dropping their labels for a moment) can be expressed in terms of derivatives of unconditional distributions,

$$\frac{\partial P(x_f|x_c)}{\partial \theta_{i\mu}} = \frac{1}{P(x_c)} \left(\frac{\partial P(x_f, x_c)}{\partial \theta_{i\mu}} - P(x_f|x_c) \frac{\partial P(x_c)}{\partial \theta_{i\mu}} \right) \tag{18}$$

in which $P(x_c) \equiv 1$ is to be substituted if $c = \emptyset$. Again, in a preprocessing step the $\{f, c\}$'s and c's that are relevant for i can be determined.

A.3 Gradient of CPTs

To proceed, we need an expression for $\partial P(x_a)/\partial \theta_{i\mu}$, where x_a plays the role of (x_f, x_c) and x_c respectively. In our parameterized Bayesian network, this probability can be expressed as

$$P(x_a) = \sum_{x'} P_{\mathrm{PType}(i)}(x'_i | x'_{\pi(i)}, \vec{\theta}_i) \prod_{j \neq i} P(x'_j | x'_{\pi(j)}) \delta_{x_a, x'_a} \tag{19}$$

in which the CPT of node i is the only term that depends on $\vec{\theta}_i$. So the derivative is

$$\frac{\partial P(x_a)}{\partial \theta_{i\mu}} = \sum_{x'} \frac{\partial P_{\mathrm{PType}(i)}(x'_i | x'_{\pi(i)}, \vec{\theta}_i)}{\partial \theta_{i\mu}} \prod_{j \neq i} P(x'_j | x'_{\pi(j)}) \delta_{x_a, x'_a} \tag{20}$$

Now we note that the functional form of the derivative of $P_{\text{PType}(i)}$ with respect to $\theta_{i\mu}$ is a property of the PType of the CPT of node i,

$$\Gamma^{\mu}_{\text{PType}}(y, y_{\pi}; \vec{\theta}) = \frac{1}{P_{\text{PType}}(y|y_{\pi}, \vec{\theta})} \frac{\partial P_{\text{PType}}(y|y_{\pi}, \vec{\theta})}{\partial \theta_{\mu}} \tag{21}$$

The gradient of the (log) CPT of PType can be stored in the library of PTypes of CPT parameterizations supplied by the tool. During the optimization it can be loaded and evaluated at $\vec{\theta}_i$. Then the derivative (20) can be expressed as

$$\frac{\partial P(x_a)}{\partial \theta_{i\mu}} = \sum_{x'_i, x'_{\pi_i}} \Gamma^{\mu}_{\text{PType}(i)}(x'_i, x'_{\pi_i}; \vec{\theta}_i) P(x'_i, x'_{\pi_i}, x_a) \tag{22}$$

which only involves a probabilistic inference computation which can be computed by our inference tool.

A.4 Full gradient

By combining (14), (16), (18) and (22), the full gradient of the cost function can be computed. The main computational cost is the computation of $P(x'_i, x'_{\pi_i}, x^{\alpha_k}_{\{f,c\}(\alpha_k)})$ for each combination of i and its relevant α_k, which is needed for (22).

Acknowledgments

This research is part of the Intelligent Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant BSIK03024.

References

[1] J. Pearl. *Probabilistic Reasoning in Intelligent systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., 1988.

[2] F.V. Jensen. *An Introduction to Bayesian networks.* UCL Press, 1996.

[3] M.J. Druzdzel and L.C. van der Gaag. Building probabilistic networks: "where do the numbers come from ?" guest editors introduction. *IEEE Transactions on Knowledge and Data Engineering*, 12:481–486, 2000.

[4] K.B. Laskey. Sensitivity analysis for probability assessments in Bayesian networks. In *UAI '93: Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence*, pages 136–142, 1993.

[5] J. Whittaker. *Graphical Models in Applied Multivariate Analysis.* Wiley, New York, 1990.

[6] W. Press, B. Flannery, A. Teukolsky, and W. Vettering. *Numerical Recipes in C.* Cambridge University Press, 1989.