Bayesian Networks

Introduction

Bayesian networks (BNs), also known as *belief networks* (or Bayes nets for short), belong to the family of probabilistic *graphical models* (GMs). These graphical structures are used to represent knowledge about an uncertain domain. In particular, each node in the graph represents a random variable, while the edges between the nodes represent probabilistic dependencies among the corresponding random variables. These conditional dependencies in the graph are often estimated by using known statistical and computational methods. Hence, BNs combine principles from graph theory, **probability theory**, computer science, and statistics.

GMs with *undirected edges* are generally called *Markov random fields* or *Markov networks*. These networks provide a simple definition of independence between any two distinct nodes based on the concept of a *Markov blanket*. Markov networks are popular in fields such as statistical physics and computer vision [1, 2].

BNs correspond to another GM structure known as a *directed acyclic graph* (DAG) that is popular in the statistics, the machine learning, and the artificial intelligence societies. BNs are both mathematically rigorous and intuitively understandable. They enable an effective representation and computation of the joint probability distribution (JPD) over a set of random variables [3].

The structure of a DAG is defined by two sets: the set of nodes (vertices) and the set of directed edges. The nodes represent random variables and are drawn as circles labeled by the variable names. The edges represent direct dependence among the variables and are drawn by arrows between nodes. In particular, an edge from node X_i to node X_j represents a statistical dependence between the corresponding variables. Thus, the arrow indicates that a value taken by variable X_j depends on the value taken by variable X_i, or roughly speaking that variable X_i "influences" X_j. Node X_i is then referred to as a *parent* of X_j and, similarly, X_j is referred to as the *child* of X_i. An extension of these genealogical terms is often used to define the sets of "descendants" – the set of nodes that can be reached on a direct path from the node, or "ancestor" nodes – the set of nodes from which the node can be reached on a direct path [4]. The structure of the acyclic graph guarantees that there is no node that can be its own ancestor or its own descendent. Such a condition is of vital importance to the factorization of the joint probability of a collection of nodes as seen below. Note that although the arrows represent direct causal connection between the variables, the *reasoning process* can operate on BNs by propagating information in any direction [5].

A BN reflects a simple conditional independence statement. Namely that each variable is independent of its nondescendents in the graph given the state of its parents. This property is used to reduce, sometimes significantly, the number of parameters that are required to characterize the JPD of the variables. This reduction provides an efficient way to compute the posterior probabilities given the evidence [3, 6, 7].

In addition to the DAG structure, which is often considered as the "qualitative" part of the model, one needs to specify the "quantitative" parameters of the model. The parameters are described in a manner which is consistent with a Markovian property, where the conditional probability distribution (CPD) at each node depends only on its parents. For discrete random variables, this conditional probability is often represented by a table, listing the local probability that a child node takes on each of the feasible values – for each combination of values of its parents. The joint distribution of a collection of variables can be determined uniquely by these local conditional probability tables (CPTs).

Following the above discussion, a more formal definition of a BN can be given [7]. A Bayesian network B is an annotated acyclic graph that represents a JPD over a set of random variables $\mathbf{V}$. The network is defined by a pair $B = \langle G, \Theta \rangle$, where G is the DAG whose nodes $X_1, X_2, \ldots, X_n$ represents random variables, and whose edges represent the direct dependencies between these variables. The graph G encodes independence assumptions, by which each variable X_i is independent of its nondescendents given its parents in G. The second component Θ denotes the set of parameters of the network. This set contains the parameter $\theta_{x_i|\pi_i} = P_B(x_i|\pi_i)$ for each realization x_i of X_i conditioned on π_i, the set of parents of X_i in G. Accordingly, B defines a unique JPD over $\mathbf{V}$, namely:

$$P_B(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i|\pi_i) = \prod_{i=1}^{n} \theta_{X_i|\pi_i} \tag{1}$$

For simplicity of representation we omit the subscript B henceforth. If X_i has no parents, its local probability distribution is said to be *unconditional*, otherwise it is *conditional*. If the variable represented by a node is *observed*, then the node is said to be an evidence node, otherwise the node is said to be hidden or latent.

Consider the following example that illustrates some of the characteristics of BNs. The example shown in Figure 1 has a similar structure to the classical "earthquake" example in Pearl [3]. It considers a person who might suffer from a back injury, an event represented by the variable *Back* (denoted by B). Such an injury can cause a backache, an event represented by the variable *Ache* (denoted by A). The back injury might result from a wrong sport activity, represented by the variable *Sport* (denoted by S) or from new uncomfortable chairs installed at the person's office, represented by the variable *Chair* (denoted by C). In the latter case, it is reasonable to assume that a coworker will suffer and report a similar backache syndrome, an event represented by the variable *Worker* (denoted by W). All variables are binary; thus, they are either true (denoted by "T") or false (denoted by "F"). The CPT of each node is listed besides the node.

In this example the parents of the variable *Back* are the nodes *Chair* and *Sport*. The child of *Back* is *Ache*, and the parent of *Worker* is *Chair*. Following the BN independence assumption, several independence statements can be observed in this case. For example, the variables *Chair* and *Sport* are marginally independent, but when *Back* is given they are conditionally dependent. This relation is often called *explaining away*. When *Chair* is given, *Worker* and *Back* are conditionally independent. When *Back* is given, *Ache* is conditionally independent of its ancestors *Chair* and *Sport*. The conditional independence statement of the BN provides a compact factorization of the JPDs. Instead of factorizing the joint distribution of all the variables by the chain rule, i.e., P(C,S,W,B,A) = P(C)P(S|C)P(W|S,C)P(B|W,S,C)P(A|B,W,S,C), the BN defines a unique JPD in a factored form, i.e. P(C,S,W,B,A) = P(C)P(S)P(W|C)P(B|S, C)P(A|B). Note that the BN form reduces the number of the model parameters, which belong to a multinomial distribution in this case, from $2^5 - 1 = 31$ to 10 parameters. Such a reduction provides great benefits from inference, learning (parameter **estimation**), and computational perspective. The resulting model is more robust with respect to **bias**-variance effects [8]. A practical graphical criterion that helps to

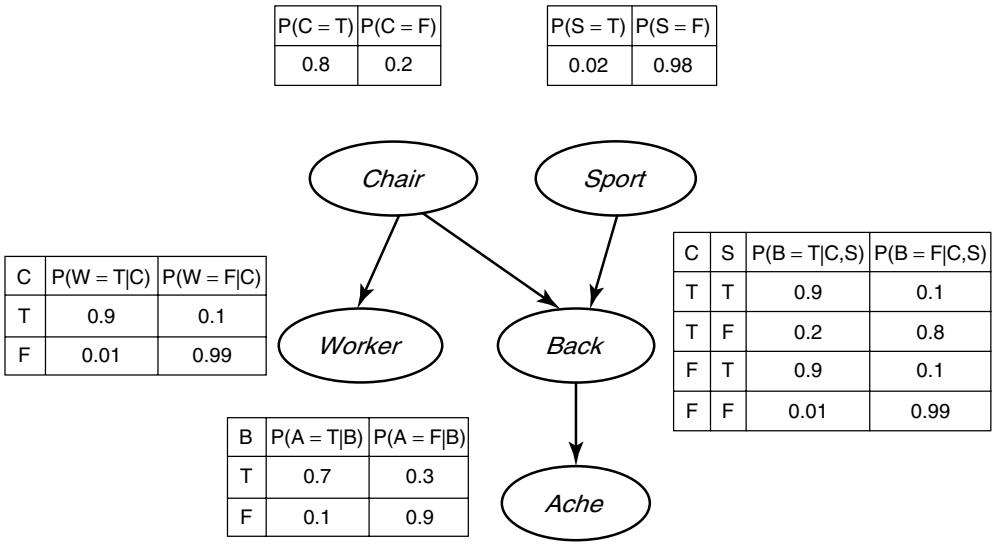

Figure 1 The backache BN example

investigate the structure of the JPD modeled by a BN is called *d-separation* [3, 9]. It captures both the conditional independence and dependence relations that are implied by the Markov condition on the random variables [2].

Inference via BN

Given a BN that specified the JPD in a factored form, one can evaluate all possible inference queries by marginalization, i.e. summing out over "irrelevant" variables. Two types of inference support are often considered: *predictive support* for node X_i, based on evidence nodes connected to X_i through its parent nodes (also called *top-down reasoning*), and *diagnostic support* for node X_i, based on evidence nodes connected to X_i through its children nodes (also called *bottom-up reasoning*). Given the example in Figure 1, one might consider the diagnostic support for the belief on new uncomfortable chairs installed at the person's office, given the observation that the person suffers from a backache. Such a support is formulated as follows:

$$\mathrm{P(C = T|A = T)} = \frac{\mathrm{P(C = T,\ A = T)}}{\mathrm{P(A = T)}} \qquad (2)$$

where

$$\mathrm{P(C = T,\ A = T)} = \sum_{\mathrm{S,W,B}\in\{\mathrm{T,F}\}} \mathrm{P(C = T)P(S)} \times \mathrm{P(W|C = T)P(B|S,\ C = T)P(A = T|B)} \qquad (3)$$

and

$$\mathrm{P(A = T)} = \sum_{\mathrm{S,W,B,C}\in\{\mathrm{T,F}\}} \mathrm{P(C)P(S)P(W|C)P(B|S,C)} \times \mathrm{P(A = T|B)} \qquad (4)$$

Note that even for the binary case, the JPD has size $O(2^n)$, where n is the number of nodes. Hence, summing over the JPD takes exponential time. In general, the full summation (or integration) over discrete (continuous) variables is called *exact inference* and known to be an *NP-hard problem*. Some efficient algorithms exist to solve the exact inference problem in restricted classes of networks. One of the most popular algorithms is the *message passing algorithm* that solves the problem in $O(n)$ steps (linear in the number of nodes) for *polytrees* (also called *singly connected networks*), where there is at most one path between any two nodes [3, 5]. The algorithm was extended to general networks by Lauritzen and Spiegelhalter [10]. Other exact inference methods include the *cycle-cutset* conditioning [3] and variable elimination [11].

Approximate inference methods were also proposed in the literature, such as ***Monte Carlo*** sampling that gives gradually improving estimates as sampling proceeds [9]. A variety of standard ***Markov chain Monte Carlo*** (MCMC) methods, including the *Gibbs sampling* and the *Metropolis–Hastings algorithm*, were used for approximate inference [4]. Other methods include the *loopy belief propagation* and *variational methods* [12] that exploit the **law of large numbers** to approximate large sums of random variables by their means.

BN Learning

In many practical settings the BN is unknown and one needs to learn it from the data. This problem is known as the *BN learning problem*, which can be stated informally as follows: Given training data and prior information (e.g., **expert knowledge**, **casual relationships**), estimate the graph topology (network structure) and the parameters of the JPD in the BN.

Learning the BN structure is considered a harder problem than learning the BN parameters. Moreover, another obstacle arises in situations of *partial observability* when nodes are hidden or when data is missing. In general, four BN learning cases are often considered, to which different learning methods are proposed, as seen in Table 1 [13].

In the first and simplest case the goal of learning is to find the values of the BN parameters (in each CPD) that maximize the (log)likelihood of the training

Table 1 Four cases of BN learning problems

Case	BN structure	Observability	Proposed learning method
1	Known	Full	Maximum-likelihood estimation
2	Known	Partial	EM (or gradient ascent), MCMC
3	Unknown	Full	Search through model space
4	Unknown	Partial	EM + search through model space

dataset. This dataset contains m cases that are often assumed to be independent. Given training dataset $\mathbf{\Sigma} = \{\mathbf{x_1}, \ldots, \mathbf{x_m}\}$, where $\boldsymbol{x_l} = (x_{l1}, \ldots, x_{ln})^T$, and the parameter set $\mathbf{\Theta} = (\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_n})$, where $\boldsymbol{\theta}_i$ is the vector of parameters for the conditional distribution of variable X_i (represented by one node in the graph), the log-likelihood of the training dataset is a sum of terms, one for each node:

$$\log L(\Theta|\Sigma) = \sum_m \sum_n \log P(x_{li}|\pi_i, \theta_i) \qquad (5)$$

The log-likelihood scoring function *decomposes* according to the graph structure; hence, one can maximize the contribution to the log-likelihood of each node independently [14]. Another alternative is to assign a prior **probability density function** to each parameter vector and use the training data to compute the posterior parameter distribution and the Bayes estimates. To compensate for zero occurrences of some sequences in the training dataset, one can use appropriate (mixtures of) conjugate prior distributions, e.g. the Dirichlet prior for the multinomial case as in the above backache example or the Wishart prior for the Gaussian case. Such an approach results in a maximum *a posteriori* estimate and is also known as the *equivalent* ***sample size*** (ESS) method.

In general, the other learning cases are computationally intractable. In the second case with known structure and partial observability, one can use the **EM (expectation maximization) algorithm** to find a locally optimal maximum-likelihood estimate of the parameters [4]. MCMC is an alternative approach that has been used to estimate the parameters of the BN model. In the third case, the goal is to learn a DAG that best explains the data. This is an NP-hard problem, since the number of DAGs on N variables is superexponential in N. One approach is to proceed with the simplest assumption that the variables are conditionally independent given a class, which is represented by a single common parent node to all the variable nodes. This structure corresponds to the *naïve BN*, which surprisingly is found to provide reasonably good results in some practical problems. To compute the Bayesian score in the fourth case with partial observability and unknown graph structure, one has to marginalize out the hidden nodes as well as the parameters. Since this is usually intractable, it is common to use an asymptotic approximation to the posterior called *Bayesian information criterion* (BIC) also known as the *minimum description length* (MDL) approach. In this case one considers the trade-off effects between the likelihood term and a penalty term associated with the model complexity. An alternative approach is to conduct local search steps inside of the M step of the EM algorithm, known as *structural EM*, that presumably converges to a local maximum of the BIC score [7, 13].

BN and Other Markovian Probabilistic Models

It is well known that classic machine learning methods like Hidden Markov models (HMMs), **neural networks**, and Kalman filters can be considered as special cases of BNs [4, 13] Specific types of BN models were developed to address stochastic processes, known as *dynamic BN*, and counterfactual information, known as *functional BN* [5]. Ben-Gal *et al.* [8] defined a hierarchical structure of Markovian GMs, which we follow here. The structure is described within the framework of DNA sequence classification, but is relevant to other research areas. The authors introduce the *variable-order Bayesian network* (VOBN) model as an extension of the *position weight matrix* (PWM) model, the *fixed-order Markov model* (MM) including HMMs, the *variable-order Markov* (VOM) model, and the BN model.

The PWM model is presumably the simplest and the most common context-independent model for DNA sequence classification. The basic assumption of the PWM model is that the random variables (e.g., nucleotides at different positions of the sequence) are statistically independent. Since this model has no memory it can be regarded as a fixed-order MM of order 0. In contrast, higher fixed-order models, such as MMs, HMMs, and interpolated MMs, rely on the statistical dependencies within the data to indicate repeating motifs in the sequence.

VOM models stand in between the above two types of models with respect to the number of model parameters. In fact, VOM models do not ignore statistical dependencies between variables in the sequence, yet, they take into account only those dependencies that are statistically significant. In contrast to fixed-order MMs, where the order is the same for all positions and for all contexts, in VOM models the order may vary for each position, based on its contexts.

Unlike the VOM models, which are homogeneous and which allow statistical dependences only between

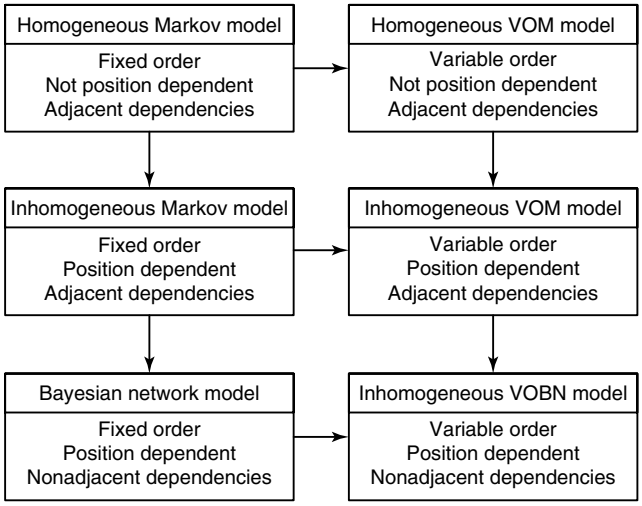

Figure 2 Hierarchical structure of Markovian graphical models [© OUP, 2005.]

adjacent variables in the sequence, VOBN models are inhomogeneous and allow statistical dependences between nonadjacent positions in a manner similar to BN models. Yet, as opposed to BN models, where the order of the model at a given node depends only on the size of the set of its parents, in VOBN models the order also depends on the context, i.e. on the specific observed realization in each set of parents. As a result, the number of parameters that need to be estimated in VOBN models is potentially smaller than in BN models, yielding a smaller chance for overfitting of the VOBN model to the training dataset. Context-specific BNs (e.g., [15, 16]) are closely related to, yet constructed differently from, the VOBN models [8].

To summarize, the VOBN model can be regarded as an extension of PWM, fixed-order Markov, and BN models as well as VOM models in the sense that these four models are special cases of the VOBN model. This means that in cases where statistical dependencies are insignificant, the VOBN model degenerates to the PWM model. If statistical dependencies exist only between adjacent positions in the sequence and the memory length is identical for all contexts, the VOBN model degenerates to an inhomogeneous fixed-order MM. If, in addition, the CPDs are identical for all positions, the VOBN model degenerates to a homogeneous fixed-order MM. If the memory length for a given position is identical for all contexts and depends only on the number of parents, the VOBN model degenerates to a BN model. If the context-dependent statistical dependencies in the VOBN model are restricted to adjacent positions, the VOBN model degenerates to the inhomogeneous VOM model. If, in addition, the context-dependent CPDs are identical for all positions, the VOBN model degenerates to a homogeneous VOM model. Figure 2 sketches these relationships between fixed-order MMs, BN models, VOM models, and VOBN models.

Summary

BNs became extremely popular models in the last decade. They have been used for applications in various areas, such as machine learning, text mining, natural language processing, speech recognition, signal processing, bioinformatics, error-control codes, medical diagnosis, weather forecasting, and cellular networks.

The name BNs might be misleading. Although the use of Bayesian statistics in conjunction with BN provides an efficient approach for avoiding data overfitting, the use of BN models does not necessarily imply a commitment to Bayesian statistics. In fact, practitioners often follow frequentists' methods to estimate the parameters of the BN. On the other hand, in a general form of the graph, the nodes can represent not only random variables but also

hypotheses, beliefs, and latent variables [13]. Such a structure is intuitively appealing and convenient for the representation of both causal and probabilistic semantics. As indicated by David [17], this structure is ideal for combining prior knowledge, which often comes in causal form, and observed data. BN can be used, even in the case of missing data, to learn the causal relationships and gain an understanding of the various problem domains and to predict future events.

Acknowledgment

The author would like to thank Prof. Yigal Gerchak for reviewing the final manuscript.

References

[1] Jordan, M.I. (1999). *Learning in Graphical Models*, MIT Press, Cambridge.

[2] Stich, T. (2004). Bayesian networks and structure learning, Diploma Thesis, Computer Science and Engineering, University of Mannheim, available at: http://66.102.1.104/scholar?hl=en&lr=&q=cache:j36KPn-8hWroJ:www.timostich.de/resources/thesis.pdf.

[3] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Francisco.

[4] Griffiths, T.L. & Yuille, A. (2006). A primer on probabilistic inference, *Trends in Cognitive Sciences* Supplement to special issue on Probabilistic Models of Cognition, **10**(7), 1–11.

[5] Pearl, J. & Russel, S. (2001). Bayesian networks. Report (R-277), November 2000, in *Handbook of Brain Theory and Neural Networks*, M. Arbib, ed, MIT Press, Cambridge, pp. 157–160.

[6] Spirtes, P., Glymour, C. & Schienes, R. (1993). *Causation Prediction and Search*, Springer-Verlag, New York.

[7] Friedman, N., Geiger, D. & Goldszmidt, M. (1997). Bayesian network classifiers, *Machine Learning* **29**, 131–163.

[8] Ben-Gal, I., Shani, A., Gohr, A., Grau, J., Arviv, S., Shmilovici, A., Posch, S. & Grosse, I. (2005). Identification of transcription factor binding sites with variable-order Bayesian networks, *Bioinformatics* **21**(11), 2657–2666.

[9] Pearl, J. (1987). Evidential reasoning using stochastic simulation of causal models, *Artificial Intelligence* **32**(2), 245–258.

[10] Lauritzen, S.L. & Spiegelhalter, D.J. (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society. Series B* **50**(2), 157–224.

[11] Zhang, N.L. & Poole, D. (1996). Exploiting causal independence in Bayesian network inference, *Journal of Artificial Intelligence Research* **5**, 301–328.

[12] Jordan, M.I., Ghahramani, Z., Jaakkola, T.S. & Saul, L.K. (1998). An introduction to variational methods for graphical models, in *Learning in Graphical Models*, M.I. Jordan, ed, Kluwer Academic Publishers, Dordrecht.

[13] Murphy, K. (1998). *A brief introduction to graphical models and Bayesian networks*. http://www.cs.ubc.ca/∼murphyk/Bayes/bnintro.html.
Earlier version appears at Murphy K. (2001) The Bayes Net Toolbox for Matlab, Computing Science and Statistics, 33, 2001.

[14] Aksoy, S. (2006). *Parametric Models: Bayesian Belief Networks*, Lecture Notes, Department of Computer Engineering Bilkent University, available at http://www.cs.bilkent.edu.tr/∼saksoy/courses/cs551/slides/cs551_parametric4.pdf.

[15] Boutilier, C., Friedman, N., Goldszmidt, M. & Koller, D. (1996). Context-specific independence in Bayesian networks, in *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, Portland, August 1–4 1996, pp. 115–123.

[16] Friedman, N. & Goldszmidt, M. (1996). Learning Bayesian networks with local structure, in *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, Portland, August 1–4 1996.

[17] David, H. (1999). A tutorial on learning with Bayesian networks, in *Learning in Graphical*, M.J. Models, ed, MIT Press, Cambridge, Also appears as Technical Report MSR-TR-95-06, Microsoft Research, March, 1995. An earlier version appears as Bayesian Networks for Data Mining, Data Mining and Knowledge Discovery, 1:79–119, 1997.

Further Reading

Geman, S. & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 721–741.

Metropolis, A.W., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. & Teller, E. (1953). Equations of state calculations by fast computing machines, *Journal of Chemical Physics* **21**, 1087–1092.

Tenenbaum, J.B., Griffiths, T.L. & Kemp, C. (2006). Theory-based Bayesian models of inductive learning and reasoning, *Trends in Cognitive Science* **10**, 309–318.

Related Article

Bayesian Networks in Reliability.

IRAD BEN-GAL