Advances in Bayesian Learning

Irina Rish
IBM T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY, 10532, U.S.A.
rish@us.ibm.com

Abstract

Bayesian learning is a probabilistic approach to building models that combine prior knowledge with new information extracted from data. In the past few years, significant progress has been made in learning graphical models *such as* Bayesian networks*. Bayesian networks provide a compact representation for complex multivariate distributions and accommodate efficient inference algorithms. Bayesian networks have been successfully used in many practical applications including medical diagnosis, troubleshooting in computer systems, traffic control, signal processing, bio-informatics and web data analysis. This paper provides a brief overview of state-of-the-art approaches to inference and learning in Bayesian networks and discusses further research opportunities.*

Keywords: Bayesian networks, learning, inference

1 Introduction

Bayesian approach has a long successful history in statistics, machine learning and pattern recognition (see, for example, [14] for a classical introduction into Bayesian techniques). Bayesian reasoning is a probabilistic approach to inference based on combining prior knowledge with observed data using *Bayes' rule*:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)}, \qquad (1)$$

where $P(H)$ is the *prior* probability of hypothesis H, $P(D)$ is the prior probability of observing data D, $P(D|H)$ (called *likelihood*) is the probability of observing D if hypothesis H holds, and $P(H|D)$ is the *posterior* probability of H after observing data D.

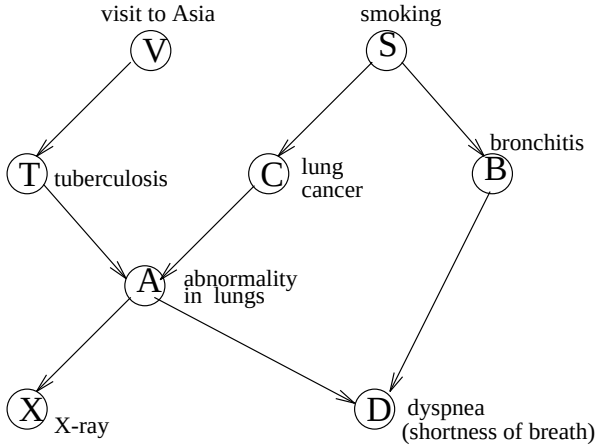

Figure 1: An example of a Bayesian network.

An increasingly popular framework for Bayesian reasoning are *graphical models* such as *Bayesian networks*. *Bayesian networks* (also called *belief networks*) provide a graphical framework for compact representation of multivariate probabilistic distributions and for efficient reasoning techniques. A Bayesian network is a directed acyclic graph, where the nodes represent random variables of interest (e.g., the temperature of a device, the gender of a patient, a feature of an object, an occurrence of an event) and the edges denote probabilistic dependencies. Since the directed edges are often interpreted as direct causal influences between the variables, Bayesian networks are also called *causal networks*. Figure 1 shows a sample Bayesian network for medical diagno-

sis, which relates diseases, such as tuberculosis or lung cancer, to possible causes (e.g., smoking) and to the symptoms (shortness of breath) and test results (X-ray).

Bayesian networks are traditionally used for representing uncertain expert knowledge and for subsequent reasoning under uncertainty in various applications, including medical diagnosis, computer troubleshooting, traffic control, speech recognition, and error-correcting codes, among others. A variety of efficient reasoning algorithms for inference in Bayesian networks has been developed over the last decade. More recently, especially with increasing volumes of data available in biomedical, Internet, and e-business applications, the research focus is shifting more towards learning Bayesian networks from data. In the past few years, significant progress has been made in developing techniques for Bayesian learning [16]. Examples include surprisingly successful applications of simple *naive Bayes* model for classification tasks in real-life applications, such as text categorization [20], and its extensions to more complex *tree-augmented naive Bayes* [15].

Learning in Bayesian networks has several advantages. First of all, Bayesian networks provide a natural framework for combining prior (expert) knowledge with learning from data. Also, learning in Bayesian networks is incremental and does not require eliminating a candidate hypothesis if it is found inconsistent with an example; rather, its probability is decreased (or increased, if the example supports the hypothesis). Second, Bayesian networks offer a better way of handling missing data than some standard statistical techniques (e.g. regression), since they encode dependencies among all variables and allow for techniques such as the *expectation-maximization (EM)* algorithm for handling incomplete data sets. Third, since directed edges often have causal interpretation, Bayesian networks can be used for learning causal relationship rather than simply dependencies among variables.

In this paper, we provide a brief introduction into existing techniques for inference and learning in Bayesian networks. The paper is organized as follows. Section 2 gives formal definitions of Bayesian networks and describes probabilistic inference algorithms based on *variable-elimination* approach. Section 3 discusses learning in Bayesian networks, which can be separated into learning the graph structure and learning the parameters of the network. Both learning from complete data and learning from data with missing values are considered. Section 4 concludes the paper with a summary and a discussion of future work.

2 Inference in Bayesian networks

Let $\mathbf{X} = \{X_1, ..., X_n\}$ be a set of random variables each having a set of possible *states*, or *values*, denoted $Val(X_i)$. Capital letters (with indexes) such as X_i will denote variables and lower-case letters such as x_i will denote their values. Boldface letters denote sets, e.g. $\mathbf{X}$ denotes the set of variables, while $\mathbf{x}$ denotes the corresponding assignment to the variables in $\mathbf{X}$. A *directed graph* is a pair $G = \{\mathbf{X}, \mathbf{E}\}$, where $\mathbf{X} = \{X_1, ..., X_n\}$ is a set of nodes and $\mathbf{E} = \{(X_i, X_j) | X_i, X_j \in X, i \neq j\}$ is a set of edges. A directed graph is called *acyclic* if it has no directed cycles. Given an edge $(X_i, X_j) \in \mathbf{E}$; X_i is called a *parent* of X_j, while X_j is called a *child* of X_i. The set of parent nodes of X_i is denoted $\mathbf{Pa_i}$. A node and its parents will be called a *family*.

A Bayesian network is a pair (G, Θ), where $G = (\mathbf{X}, \mathbf{E})$ is a directed acyclic graph representing the variables in $\mathbf{X}$ as nodes and $\Theta = \{\theta_{x_i, \mathbf{pa_i}}\}$ is the set of parameters that represent conditional probabilities for each node given its parents in G, i.e. $\theta_{x_i, \mathbf{pa_i}} = P(X_i = x_i | \mathbf{Pa_i} = \mathbf{pa_i})$ (or, using a shorter notation, $P(x_i | \mathbf{pa_i})$). The distributions $P(X_i | \mathbf{Pa_i})$, associated with each node X_i (as shown in Figure 2a), are called *local probability distributions* [16]. Typically, Bayesian networks are defined for discrete variables with finite number of states. Thus, the local probability distributions are represented by $(k+1)$-dimensional *conditional probability tables (CPTs)*, where k

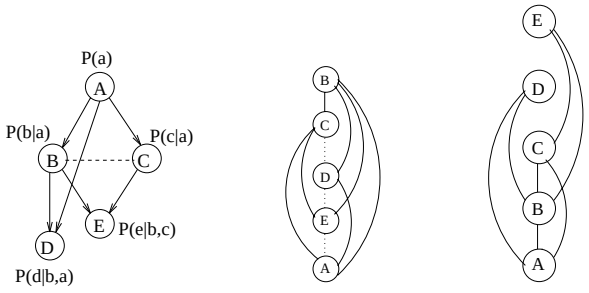

Figure 2: (a) A belief network, (b) its induced graph along $o = (A, E, D, C, B)$, and (c) its induced graph along $o = (A, B, C, D, E)$.

is the number of parents, and each entry $\theta_{x_i,\mathbf{pa_i}}$ corresponds to a particular value assignment to X_i and its parents. A Bayesian network represents a joint probability distribution over $\mathbf{X}$ as a product of local distributions:

$$P(x_1, ..., x_n) = \Pi_{i=1}^{n} P(x_i|\mathbf{pa_i}) \qquad (2)$$

The graph G^M obtained from G by connecting ("marrying") all the parents of each node (e.g., the dotted line in Figure 2a represents such an edge) and removing the directionality of edges is called the *moral graph* of Bayesian network. The moral graph is used by inference algorithms. The set of arguments of a function f is called the *scope* of f. Thus the scope of a CPT is its family, which corresponds to a clique in a moral graph.

Probabilistic inference, or *belief updating*, in Bayesian networks is defined as finding the posterior probability of a subset of variables $\mathbf{Y} \subseteq \mathbf{X}$ given the observed values of some other variables (called *evidence*). For example, using the network in Figure 1, we can assess the probability that a patient has lung cancer given that he suffers from dyspnoea (shortness of breath), but has normal X-ray results and does not smoke.

Probabilistic inference in Bayesian networks is NP-hard [7]. However, there exists a polynomial *belief-propagation* algorithm for *polytrees*, also called *singly-connected* networks [21] (i.e., the networks without undirected cycles). The two main approaches to extending this algorithm to general (multiply-connected) networks are the *cycle-cutset* approach, also called *conditioning*, and *join-tree* algorithm [21, 19, 25], which is also closely related to *variable-elimination* techniques [8, 27, 10]. When exact inference is intractable, various approximation methods, such as Monte Carlo methods, variational approximation, and local inference algorithms can be used.

The following example illustrates a variable-elimination approach on a simple network shown in Figure 2a. Assume that we want to update the belief in A given the evidence $E = 0$, i.e. to find $P(a|e = 0) = \frac{P(a,e=0)}{P(e=0)}$. It is enough to compute $P(a, e)$ since $1/P(e = 0)$ is a normalizing constant independent of a. Assuming a *variable ordering* $o = A$, E, D, C, B, we get

$$P(a, e = 0) = \sum_{e=0,d,c,b} P(a, b, c, d, e) =$$

$$\sum_{e=0,d,c,b} P(a)P(c|a)P(e|b, c)P(d|a, b)P(b|a).$$

By the distributivity law, the summation over each variable can be "pushed" to the right, as far as possible, yielding the following expression for $P(a, e = 0)$:

$$P(a) \sum_{e=0} \sum_{d} \sum_{c} P(c|a) \sum_{b} P(e|b, c)P(d|a, b)P(b|a). \qquad (3)$$

We can compute the sum in equation (3) sequentially, from right to left, by *eliminating variables* (summing over them) from the last to the first along the ordering o:
B: $h^B(a, d, c, e) = \sum_b P(e|b, c)P(d|a, b)P(b|a)$
C: $h^C(a, d, e) = \sum_c P(c|a)h^B(a, d, c, e)$
D: $h^D(a, e) = \sum_d h^C(a, d, e)$
E: $h^E(a) = h^d(a, E = 0)$
A: $Bel(a) = P(a|E = 0) = \alpha P(a)h^E(a)$,
where α is a normalizing constant, and h^X is an intermediate result of computation after summing over the variable X and its successors in o. One of the most recent examples of the variable-elimination approach to belief updating is algorithm *elim-bel* [10], a *bucket-elimination* scheme that uses the data structure called *buckets* for keeping the functions

associated with each variable. Since the table representation (e.g., CPT) of a function defined on k variables has $O(exp(k))$ entries, the time and space complexity of variable-elimination is exponential in the scope size of largest function recorded. Recording a new function amounts to *inducing* edges in the network among the variables in the scope of the function (Figure 2b shows the induced edges as dotted lines). Thus, variable elimination along a particular variable ordering results into the corresponding *induced graph*. It was shown in [11] that time and space complexity of variable elimination is exponential in the graph parameter called *induced width* [12] which describes the largest clique in the induced graph, and which corresponds to the largest scope of function recorded by the algorithm. The induced width will vary depending on the variable ordering. For example, Figures 2b and 2c depict ordered *induced graphs* obtained from the moral graph of the network in Figure 2a along the orderings $o = (A, E, D, C, B)$ and $o' = (A, B, C, D, E)$, respectively. Clearly, $w_o^* = 4$ and $w_{o'}^* = 2$. Although finding a minimum-w^* ordering is NP-hard [1], good heuristic algorithms are available; for more details on bucket-elimination and induced width see [11, 9].

3 Learning Bayesian networks

In this section, we give an overview of the state-of-the-art methods for learning Bayesian networks from data. Learning a Bayesian network is an *unsupervised learning* problem which can be informally stated as follows: given a set of observations $D = \{\mathbf{y^1}, \ldots, \mathbf{y^N}\}$ (training data), find a Bayesian network (G, Θ) that best matches D. This is an optimization problem with respect to a particular *scoring function* defined on a Bayesian network.

Popular scoring functions include the *Bayesian score* [6, 17], the *Minimum Description Length (MDL)* criterion [4, 18], and the equivalent to it *Bayesian information criterion (BIC)* [24] (often called BIC/MDL). These scoring functions are asymptotically equivalent and asymptotically correct, i.e., as the number of samples increases, the distribution encoded by the learned network converges with probability 1 to the true distribution the observations were sampled from [16, 3]. We will focus on the commonly used BIC/MDL score.

MDL [22] is an information-theoretic criterion that favors models that provide the shortest description of the training data. This description includes both the description of the model and the description of the data given the model. Formally, given a Bayesian network $BN = (G, \Theta)$, and a training data set D, the MDL score of BN is defined as

$$MDL(G, \Theta|D) = \frac{logN}{2}|\Theta| - logP(D|\Theta, G), \tag{4}$$

where $|\Theta|$ is the number of parameters in the network. Without going into details of MDL derivation, we just note here that the first term of the MDL score is the description length of a Bayesian network, i.e. the number of bits required to encode the network parameters (each parameter can be encoded using $\frac{logN}{2}$ bits), while the second term, the negative log-likelihood of the model BN given data D, gives the number of bits needed to describe D when using BN.

Minimizing MDL is equivalent to maximizing another well-known score, the *Bayesian information criterion (BIC)* [24], which is exactly the negation of MDL. Intuitively, both MDL and BIC favor models that predict data better (have higher log-likelihood $logP(D|\Theta, G)$) and have lower representation complexity ($\frac{logN}{2}|\Theta|$).

Thus, learning a Bayesian network from data D can be formally stated as finding

$$(G, \Theta) = \arg\min_{G,\Theta} MDL(G, \Theta). \tag{5}$$

A common approach to this problem is to search in the space of graph structures, using $MDL(G) = min_\Theta MDL(G, \Theta)$ score for each graph. Since the first term in the equation 4 does not depend Θ, we only need to minimize the second one, i.e. to find Θ maximizing

the log-likelihood $logP(D|\Theta, G)$ given D and G. This task is addressed in the following section.

3.1 Learning parameters

When learning parameters Θ for a given graph structure G, the following two cases are usually considered: the case of *complete* training data (i.e., all variables are observed), and the case of *missing data* (i.e., some of the variables may have missing values). The first problem is straightforward statistical parameter estimation, while the second one requires more sophisticated non-linear optimization techniques.

If the training set D is *complete*, it is easy to show that the log-likelihood $logP(D|\Theta, G)$ is decomposable according to the graph structure G using the product in equation 2:

$$logP(D|\Theta, G) = \sum_{x_i, \mathbf{pa_i}} N_{x_i, \mathbf{pa_i}} log\theta_{x_i|\mathbf{pa_i}}, \quad (6)$$

where $N_{x_i, \mathbf{pa_i}}$ are *sufficient statistics* representing the number of data instances matching the instantiations $X_i = x_i$ and $\mathbf{Pa_i} = \mathbf{pa_i}$, and $\theta_{x_i|\mathbf{pa_i}}$ are the parameters of local distribution for X_i. It is easy to show that this expression is maximized by the frequencies (maximum-likelihood estimates) $\theta_{x_i, \mathbf{pa_i}} = \frac{N_{x_i, \mathbf{pa_i}}}{N_{\mathbf{pa_i}}}$, where $N_{\mathbf{pa_i}}$ is the number of samples matching the assignment $Pa_i = \mathbf{pa_i}$.

An alternative approach to parameter estimation is to use *Bayesian statistics* rather than *classical statistical approach*. Classical statistical approach assumes that parameters are fixed, although unknown, *physical probabilities*, i.e. constants that can be estimated from a set of training examples using the ML estimate. On the other hand, Bayesian statistics views the parameters as *unknown variables* governed by probability distributions. These probabilities are *subjective*, i.e. they represent our *degrees of belief*. We assume some prior belief (e.g., based on historical information) in θ that is represented by the *prior distribution* $P(\theta)$. When new data D become available, this belief is updated according to Bayes' rule $P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$. Thus, the Bayesian approach takes advantage of prior knowledge about the parameters, which is especially useful when data are scarce. How do we select prior distribution $P(\theta)$? A commonly used approach, usually motivated by computational convenience, is to use the so-called *conjugate priors*, which have the property that the posterior $P(\theta|D)$ belongs to the same *conjugate distribution family* as the prior $P(\theta)$. The conjugate family for priors is determined by the distribution family of $P(D|\theta)$. A common approach to modeling local distribution in Bayesian networks with finite-valued variables is to use the *multinomial* distribution with the parameters $\theta_{x_i, \mathbf{pa_i}}$ [16]. The conjugate prior for the multinomial distribution is the *Dirichlet distribution*:

$$Dir(\theta|\alpha_1, ..., \alpha_m) \equiv \frac{, (\alpha)}{\prod_{i=1}^{m} , (\alpha_i)} \prod_{i=1}^{m} \theta_i^{\alpha_i - 1}, \quad (7)$$

where $\alpha = \sum_{i=1}^{m} \alpha_i$ and $, (\cdot)$ is the *Gamma-function* which satisfies $, (x+1) = x, (x)$ and $, (1) = 1$. Parameters α_j are the often called *hyperparameters*, in order to distinguish them from the parameters of the corresponding multinomial distribution. Given a set of observations D_X of a multinomial m-valued variable X with the parameters $\theta = (\theta_1, ..., \theta_m)$, it is easy to see that the posterior $P(\theta|D_X)$ is also Dirichlet:

$$P(\theta|D_X) \propto P(D|\theta)P(\theta) \propto \prod_{i=1}^{m} \theta_i^{N_i} \cdot \prod_{i=1}^{m} \theta_i^{\alpha_i - 1}$$

$$\propto \prod_{i=1}^{m} \theta_i^{N_i + \alpha_i - 1},$$

and therefore (taking into account normalization constant),

$$P(\theta|D_X) = Dir(\theta|\alpha_1 + N_1, ..., \alpha_m + N_m),$$

where N_i is the number of times X had its i-th value in D_X. The parameters of the Dirichlet priors can be also interpreted as "imaginary counts" obtained from α prior observations (α is called an *equivalent sample size*). Thus, larger parameters reflect higher confidence in our prior. The *maximum a posteriori*

probability estimate of each θ_i is then

$$\theta_i^{MAP} = \frac{\alpha_i + N_i}{\alpha + N}, \qquad (8)$$

where N is the total number of samples.

Assuming multinomial distribution for each family in a Bayesian network, we can obtain the Bayesian parameter estimates (MAP) from equation 8, where $N_i = N_{x_i,\mathbf{pa_i}}$, $N = N_{\mathbf{pa_i}}$, and $\alpha_i = \alpha_{x_i,\mathbf{pa_i}}$. Note that we assumed that priors for the parameters of each family are independent, and that data samples are independent given the parameters. It is easy to show that, if there is no *missing data*, the posteriors on parameters are also independent, and thus can be computed separately for each family. For more details on conjugate priors and Bayesian parameter estimation, see [16].

We now consider the situation when some variables have missing values in D. In this case, the likelihood term in scoring functions is replaced by *marginal likelihood*, i.e. the probability of the subset of the variables that was observed in a given data instance (assuming that data are *missing at random* [23]). However, marginal likelihood does not decompose as the likelihood does. Finding $\theta_{x_i,\mathbf{pa_i}}$ that maximize marginal likelihood becomes a nonlinear optimization problem. Common approaches to this problem include gradient descent [2] and the *Expectation-Maximization (EM)* algorithm [13, 16].

As its name suggests, the EM algorithm consists of two steps, *expectation* and *maximization*. Initially, the *expectation* step assigns some (e.g., random) values to the parameters in Θ. Then, conditioned on G, Θ, and the data set D where some observations are missing, the *expected sufficient statistics* are computed:

$$E(N_{x_i,\mathbf{pa_i}}) = \sum_{k=1}^{N} P(x_i, \mathbf{pa_i}|\mathbf{y^k}, \Theta, G). \qquad (9)$$

When all variables are observed, the (empirical) probabilities in the summation above are just zero or one. When some of the variables (e.g., X_i or some of its parents Pa_i) are unobserved, the corresponding probability can be computed using any of the existing inference techniques for Bayesian networks, such as variable elimination or join-tree algorithms.

Once the expected sufficient statistics are computed, they can be used as if they were actual sufficient statistics from a complete data set D', to compute the ML estimates of the parameters Θ (*maximization step*). The ML estimates maximize $P(D'|\Theta, G)$ and can be written as

$$\theta_{x_i,\mathbf{pa_i}} = \frac{E(N_{x_i,\mathbf{pa_i}})}{\sum_{x_i \in Val(X_i)} E(N_{x_i,\mathbf{pa_i}})}. \qquad (10)$$

The expectation and maximization steps are repeated iteratively. As shown in [13], under certain regularity conditions, EM converges to a local maximum.

3.2 Learning Structure

Given a scoring function, and training data D, let us consider the problem of finding the highest-score Bayesian network among the networks in which each node has no more than k parents. In general, this is an intractable problem: it is NP-hard for $k > 1$ [5]. Therefore, a commonly used approach to this problem are heuristic search (e.g., greedy local search, best-first search) and Monte-Carlo techniques [6, 16].

Local greedy search (gradient descent) usually makes one of the following changes to the network at a time: adding an arc, deleting an arc, or reversing an arc. The network score must be re-evaluated after each change. This computation is greatly simplified if the scoring function is *separable*, i.e. it can be decomposed into a sum or a product of scoring functions restricted to each family (a variable X_i and its parents $\mathbf{Pa_i}$), which is the case for complete training data.

Another approach to learning graph structure uses *constraint-based* techniques. The independence relations inferred from data are used to constrain the space of possible graph structures [26]. Learning the graph structure is closely related to learning causal relationships, since the directed edges may have causal

semantics. The current methods for learning causal relationships are still "new and controversial" as noted in [16]. See [21, 26, 16] for an in-depth discussion on that topic.

4 Summary and discussion

Bayesian learning is a probabilistic approach to building models which combines prior knowledge with learning from data. This paper gives an overview of probabilistic inference and learning in a graphical framework called Bayesian networks which encodes probabilistic dependencies among variables of interest as a directed acyclic graph where the nodes correspond to the variables and edges represent probabilistic (and often also causal) dependencies.

Inference in Bayesian networks has been a subject of extensive research for many years. Probabilistic inference task is defined as finding the posterior probability of a subset of variables given observations. Well-known inference algorithms include *join-tree* algorithm [21, 19, 25], which is also closely related to *variable-elimination* techniques [8, 27, 10]. The complexity of such algorithms is exponential in the size of the largest clique created in a graph during inference (also called *induced width*) which corresponds to the largest function recorded by an algorithm. When the induced width is large, approximation algorithms such as Monte-Carlo techniques, variational inference, or local propagation, can be used.

Bayesian networks were traditionally used for encoding expert knowledge in expert systems. Recently, increasing volumes of data available in a variety of real-life applications, including web, e-commerce, and bioinformatics, motivated an active research on learning Bayesian networks from data. A natural way of combining prior knowledge with data is one of the major advantages of Bayesian networks. Besides, they provide a possibility for learning causal relationships, for convenient handling incomplete data, and for incremental learning, among the other advantages.

Learning a Bayesian network is typically viewed as a model selection problem based on some scoring function associated with a network. A popular score is the minimum-description length (MDL) [22], an information-theoretic criterion which casts learning as a data-compression problem and favors models that provide a shorter description of data. The scoring function is used to search in the space of graph structures. Computing this score for a particular graph requires finding a set of network parameters that best fit the data. In the presence of complete data this problem can be decomposed into simple ML parameter estimation for each local probability function. In the presence of missing data, however, the problem is not decomposable and requires non-linear optimization techniques, such as EM algorithm [13]. Common approaches to learning graph structure include heuristic search techniques and constraint-based approaches. For a survey of current approaches to learning Bayesian networks see [16].

Learning Bayesian networks is a relatively young field which has a variety of open research directions. For example, learning causal relationships is still a new (and sometimes controversial) area. While learning parameters for a fixed graph structure is a relatively well-known statistical task, better learning with incomplete data is a direction for further research (e.g., improving the efficiency of EM by using approximate inference techniques in E-step). Learning graph structure, including learning hidden variables not yet present in the graph, is even more complex and generally intractable problem requiring further investigation on better search techniques and other methods. Another important problem is learning *dynamic Bayesian networks (DBNs)* that model stochastic processes. Finally, combining Bayesian network learning with other techniques developed in statistics, machine learning, and data-mining is a promising direction for further research.

References

[1] S.A. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability - a survey. *BIT*, 25:2–23, 1985.

[2] J. Binder, D. Koller, S. Russel, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.

[3] R. Bouckaert. Properties of Bayesian network learning algorithms. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 102–109, San Francisco, CA, 1994.

[4] R. Bouckaert. Bayesian belief networks: From construction to inference. Technical report, Phd thesis, Utrecht University, Utrecht, The Netherlands, 1995.

[5] D. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: search methods and experimental results. In *Proceedings of Fifth Conference on Artificial Intelligence and Statistics*, pages 112–128, Fort Lauderdale, Florida, 1995.

[6] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[7] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405, 1990.

[8] B. D'Ambrosio. Symbolic probabilistic inference in large BN2O networks. In *Proc. Tenth Conf. on Uncertainty in Artificial Intelligence*, pages 128–135, 1994.

[9] R. Dechter. Constraint networks. In *Encyclopedia of Artificial Intelligence*, pages 276–285. John Wiley & Sons, 2nd edition, 1992.

[10] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence*, pages 211–219, 1996.

[11] R. Dechter. *Bucket elimination: A unifying framework for probabilistic reasoning.* In M. I. Jordan (Ed.), Learning in Graphical Models, Kluwer Academic Press, 1998.

[12] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1987.

[13] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[14] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis.* New York: John Wiley and Sons, 1973.

[15] N. Friedman, D. Geiger, and Goldszmidt M. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[16] D. Heckerman. A tutorial on learning Bayesian networks, technical report msr-tr-95-06. Technical report, Microsoft Research, 1995.

[17] D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[18] W. Lam and F. Bacchus. Learning Bayesian belief networks. an approach based on the mdl principle. *Computational Intelligence*, 10:269–293, 1994.

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

[20] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[21] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[22] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[23] D.R. Rubin. Inference and missing data. *Biometrica*, 63:581–592, 1995.

[24] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[25] R.D. Shachter. Evaluating influence diagrams. *Operations Research*, 34, 1986.

[26] P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search.* New York: Springer Verlag, 1993.

[27] N.L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.