An Introduction to Bayesian Networks in Systems and Control

Dr Michael Ashcroft
Computer Science Department
Uppsala University
Uppsala, Sweden
mikeashcroft@inatas.com

***Abstract*—Bayesian networks are a popular and powerful tool in artificial intelligence. They have a natural application in soft-sensing and filtering for system control. This paper provides an overview of the techniques involved. It proceeds by giving a mathematical overview of what Bayesian networks are and the flavors they come in. It then looks at how they can be created or learnt from data and the situations that lead to the use of ensemble models. Then it examines how they can be used for system analysis, inference and automated decision making. Finally, we look at their use in soft-sensing and dynamic system modeling.**

***Index Terms*—Bayesian networks, decision automation, system control, soft-sensing, stochastic modeling**

I. Introduction

Bayesian networks are a popular and powerful tool in artificial intelligence. They have a natural application in soft-sensing and filtering for system control. The point of this paper is to provide an overview of the techniques involved from this perspective. We will proceed by giving a mathematical overview of what Bayesian networks are and the flavors they come in. We then look at how they can be created or learnt from data and the situations that lead to the use of ensemble models. Then we examine how they can be used for system analysis, inference and automated decision making. Finally, we look at their use in soft-sensing and dynamic system modeling.

II. Bayesian Networks

Recall from probability theory that two random variables, X and Y, are independent if and only if $P(X,Y) = P(X)P(Y)$. Analogously, X and Y are conditionally independent given a third random variable Z if and only if $P(X,Y|Z) = P(X|Z)P(Y|Z)$, which is equivalent to:

$$P(X|Z) = P(X|Y,Z) \tag{1}$$

Also recall that the chain rule for random variables says that for n random variables, $X_1, X_2, ...X_n$, defined on the same sample space S:

$$\begin{aligned} P(X_1, X_2, ...X_n) &= P(X_n|X_{n-1}, X_{n-2}, ...X_1) \\ &\quad P(X_{n-1}|X_{n-2}, ...X_1) \\ &\quad ...P(X_2|X_1)P(X_1) \end{aligned} \tag{2}$$

Imagine we have five random variables: $\{A, B, C, D, E\}$. From the chain rule, we know that:

$$\begin{aligned} P(A,B,C,D,E) &= P(E|A,B,C,D) \\ &\quad P(D|A,B,C)P(C|A,B) \\ &\quad P(B|A)P(A) \end{aligned} \tag{3}$$

We can represent these five conditional independencies by means of a directed acyclic graph (DAG) and a set of conditional distributions, where:

- Each random variable is mapped to a node of the DAG
- Each node has associated with it the conditional distribution for its variable
- Each node has incoming edges from the nodes associated with the variables on which the node's conditional distribution is conditional

Such a representation is a Bayesian network. It satisfies the Markov conditions:

Definition A direct acyclic graph (DAG), G, with nodes N_G, a joint probability distribution, P, of random variables D_P, and a bijective mapping $f : D_P \Rightarrow N_G$ satisfies the Markov Condition if and only if for all $v \in D_P$, where $n = f(v)$, v is conditionally independent given P of the variables that are mapped to the non-descendants of n given the variables that are mapped to the parents n.

Node	Conditional Independencies
A	-
B	C and E, given A
C	B, given A
D	A and E, given B and C
E	A, B and D, given C

TABLE I: Conditional independencies required of random variables the DAG in Figure 1 to be a Bayesian Network

If we know no more than the decomposition given to us by the chain rule in equation 3, the associated Bayesian network's DAG will be complete (since each variable is conditional on all those prior to it in the decomposition order). However, imagine that we know that certain conditional independencies exist as specified in table I. From the definition of conditional independence, we know that:

- $P(C|B,A) = P(C|A)$
- $P(D|C,B,A) = P(D|C,B)$
- $P(E|D,C,B,A) = P(E|C)$

Accordingly:

$$P(A,B,C,D,E) = P(E|C)P(D|C,B)P(C|A)P(B|A)P(A) \quad (4)$$

Whenever we simply the conditional distributions in virtue of a known conditional independence relation, we remove an edge on the DAG of our Bayesian network representation. In this case, the resulting network is given by figure 1.

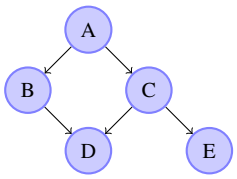

Fig. 1: A DAG with five nodes

Loosely speaking, what we have done is pull the joint probability distribution P apart by its conditional independencies. A Bayesian network is an encoding of these conditional independencies in the DAG topology coupled with the simplified conditional distributions. Note that the conditional independencies are encoded by the *absence* of edges.

The reason why Bayesian networks are useful is that this structure gives us a means of performing tractable calculations locally on the network whilst using all information of the joint distribution. It has been proven that every discrete probability distribution (and many continuous ones) can be represented by a Bayesian Network, and that every Bayesian network represents some probability distribution. Of course, if there are no conditional independencies in the joint probability distribution, representing it with a Bayesian network gains us nothing. But in practice, while independence relationship between random variables in a system we are interested in modeling are rare (and assumptions regarding such independence dangerous), conditional independencies are plentiful.

III. Discrete and continuous Bayesian networks

Bayesian networks come in a number of varieties according to the restrictions, if any, placed on the forms the conditional probability distributions can take. We will concentrate on discrete Bayesian networks, where continuous variables are discretized during preprocessing. Discrete Bayesian networks:

- Deal with continuous variables by discretization into arbitrarily many arbitrarily sized intervals. Various methods can be used for choosing the intervals, including automated clustering methods.
- Are not limited by linear and/or Gaussian noise assumptions.
- Are unrestricted by an apriori structure beyond that imposed by discretization. This is both good and bad:
  - They follow the data when it leads.
  - Cases unencountered in the learning data will take the apriori distribution, which is generally uniform. There are situations where this is undesirable (e.g. where closely related cases all strongly evince a particular structure). Methods exist that provide variance estimates which help indicate when dangerously novel cases are encountered.
- Permit efficient and accurate variance estimation on aposteriori probability distributions.
- Permit the use of exact inference algorithms.
- Permit, when combined with decision theoretic extensions, the use of exact utility maximization algorithms for generating decision policies (including on meta-models).
- Can be used as the automated basis for the production of general Bayesian networks (see below).

Other common forms include Gaussian and hybrid discrete/Gaussian networks. Automated algorithms exist for the automatic learning of, and exact inference on, such networks. These, though, require Gaussian variables to be linear combinations of their parents with Gaussian noise (potentially conditional on the values taken by their discrete in hybrid networks).

General Bayesian networks, where any conditional probability distribution in the network can be of any type, are possible. Currently, no automated learning algorithms or exact inference algorithms are known for such networks, but sampling methods do exist for inference. When such networks are desired, it has been suggested that discretized variables be used for the structural learning process [1]. After the conditional independencies are discovered by this process, bespoke conditional distributions for each variable given its parents can be fitted to the non-discretized data given domain knowledge.

Rigorous comparisons of accuracy between discrete and Gaussian networks are difficult to find. My *conjecture* is that discrete networks often offer significant advantages over their Gaussian cousins because of their non-linearity, their minimal imposition of structure on the data during learning and the current relative sophistication of the algorithms that can be perform on them. Obviously, though, the degree to which the system in question meets the assumptions involved in the Gaussian models is a key factor here. Where this is unclear but the inclusion of non-discretized continuous variables unavoidable, a good option is to custom-build a general network from the discrete conditional independence structure identified by the automated discrete learning algorithm.

IV. Learning

A. Learning a Network from Expert Causal Knowledge

Importantly, a causal network is a conditional independence encoding of the type described previously. Thus if

we have knowledge of the causal relationships pertaining between the variables we are modeling then we can immediately produce the DAG structure of the Bayesian network. In such cases, domain experts may also directly specify the conditional distributions.

Where this does not occur, we need to learn the conditional distributions from data. This is termed 'parameter learning'. In the discrete Bayesian network case, the prior conditional probability distributions associated with a node X will be a set of Dirichlet distributions, one for each possible value combination of the node's parents. These distributions are updates (in the usual way) from the training data, resulting in Dirichlet posterior distributions. We can encode prior information and/or enforce a level of conservatism through customizing the prior parameters.

B. Learning a Network from Data

We now turn to the more interesting case where no expert knowledge is involved in learning the network. The basic procedure is to perform a heuristic search on the space of possible sets of conditional independencies in order to obtain the best such set. This is complicated by the fact that multiple topologies can encode the same set of conditional independencies. Such topologies are called Markov equivalent. Since we are using heuristic methods which may become stuck at local fitness maxima, searching the DAG topologies implicitly places a prior on the sets of conditional independencies biased to those that are represented by larger numbers of DAG topologies. To overcome this, we instead search Markov equivalence classes of topologies using DAG patterns [2].

Chickering developed such an algorithm that includes an optimality guarantee: As the size of our learning data approaches infinity, the probability of learning the globally optimal model (with a single iteration of the algorithm) approaches 1 [3]. This algorithm searches over inclusion boundaries of Markov equivalence classes. Where the conditions are not met, a more general hill climb algorithm produces better results. The result is that the most robust learning algorithm utilizes the inclusion boundary algorithm for its first search and then repeatedly restarts the general hill climb algorithm.

The most principled and popular fitness function is the Bayesian Dirichlet score. This calculates the aposteriori probability of a set of conditional independencies given the learning data. Note that the use of MAP rather than ML is crucial, in that the former but not the latter implicitly penalizes complexity. It is also theoretically principled. Other fitness functions include entropy theoretic measures, such as the BIC and AIC.

At each point in the structure learning algorithms, changes in fitness associated with potential alterations to the current DAG pattern are calculated. As the utilized fitness functions are decomposable (the fitness of the DAG pattern is the sum or product of the application of the fitness function to individual conditional distributions) and score equivalent (all instances of a Markov equivalence class are equally fit), scoring proceeds by the local calculation of the results of potential alterations on a small number of affected conditional distributions in sample topologies from the equivalence classes represented by the relevant DAG patterns. The conditional distributions corresponding to a particular DAG pattern are generated on topologies from the data as described above.

The Bayesian Dirichlet fitness function is:

$$P(d|G) = \prod_{i \in G} \prod_{j \in PA(n)} \frac{\Gamma(N_{ij}^{(G)})}{\Gamma(N_{ij}^{(G)} + M_{ij}^{(G)})} \prod_{k=0}^{r} \frac{\Gamma(a_{ijk}^{(G)} + s_{ijk}^{(G)})}{\Gamma(a_{ijk}^{(G)})} \quad (5)$$

where

- d is our learning data
- G is the graph we are scoring
- n is the number of nodes in the graph
- PA is a function from a node to the possible value combinations of the parents of that node.
- $N^{(G)}$ is the sum, for graph G, of the Dirichlet prior parameters for the row of node i's conditional probability table corresponding to its parents' value combination j.
- M is the sum of the learnt additions to the Dirichlet parameters for the same row.
- r_i is the number of values node i has.
- a is the Dirichlet prior parameter corresponding to value k in row j for node i in graph G.
- s is the sum of the learnt additions to the same parameter.

To ensure that the Bayesian Dirichlet fitness function is score equivalent, we must use an equivalent sample size. What this means is that we pick some number, n, and fix it such that the prior parameters in the Dirichlet distributions associated with each nodes conditional probability table sum to n.

Computational simplification (and, potentially, additional information about the actual system being modeled) can be obtained by permitting the introduction, during structure learning, of new 'latent' variables where these increase the fitness of the resulting network. These act as intermediaries between sets of parents and children.

Structural learning can be performed with missing data items in the learning set. Common algorithms for dealing with this are the Gibbs Sampler and Expectation Maximization.

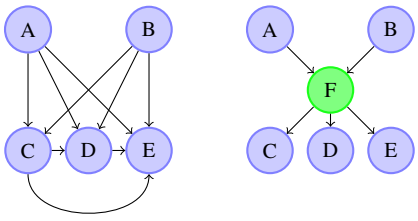

Fig. 2: An example situation where a latent variable, F, might be 'found'.

C. Meta-Models

Often a single network structure dominates alternatives. Where this is not the case, we can collect multiple high

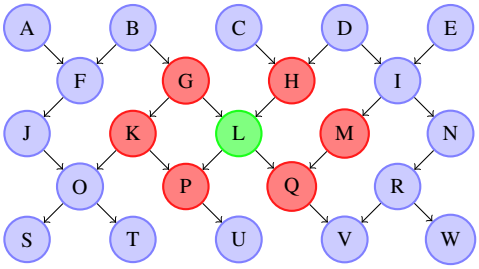

Fig. 3: The Markov Blanket of Node L

scoring networks by, for example, collecting all networks that are at least $\frac{1}{x}$ as probable (assuming the use of the Bayesian Dirichlet fitness function) as the best network, for some x. These networks can be weighted by their relative probability and inference can be performed over the entire set. Effectively, we now reason using not just our best hypothesis of the system structure, but a set of plausible hypotheses, weighted for their plausibility. This can be a very powerful method, and all inference algorithms discussed below can be run on such a meta-model.

V. System Analysis

A. Markov Blanket

The Markov Condition entails other conditional independencies. Because of the Markov Condition, these conditional independencies have a graph theoretic criterion called D-Separation (see [4] for detailed definition). Accordingly, when one set of random variables, Γ, is conditionally independent of another, Δ, given a third, Θ, them we will say that the nodes representing the random variables in Γ are D-Separated from Δ by Θ.

The most important case of D-Separation/Conditional Independence is:

- A node is D-Separated of the entire graph given its parents, its children, and the other parents of its children.

Because of this, the parents, children and other parents of a node's children are called the *Markov Blanket* of the node.

So if we have a variable, α, whose probability distribution we wish to predict and whose Markov Blanket is the set of nodes, Γ. If we know the value of every node in Γ, then we know that there is no more information regarding the value taken by α.

In this way, if we are confident that we can always establish the values of some of the variables our network is modeling, we can often see that some of the remaining variables are superfluous, and we need not continue to include them in the network nor collect information on them. Since, in practice, collecting data on random variables can be costly, this can be very helpful.

B. Causal Analysis

The connection between causality and conditional independence has lead to the use of Bayesian networks in causal analysis, often in conjunction with manipulation tests. See [4] for details.

VI. Inference

Inference is the practice of obtaining aposteriori probability distributions for variables of interest, given the available evidence. As noted earlier, Bayesian networks perform tractable inference using the information of the full joint probability distribution of a system through exploiting the decomposition of that distribution into conditional probability distributions simplified by found conditional independencies and then performing local calculations on the resulting structure. Both exact and sampling algorithms exist, and the interested readers can consult [4], [5] and [6].

VII. Automated Decision Making

Bayesian networks can be extended with utility, decision and information nodes to produce 'Influence Diagrams'. The utilities are entered by domain experts and specify the value to the user of the system being in particular states. Variables under the user's control are designated decision variables. Additionally an information order is stipulated. This is based on the partial order in which the decisions must be made as well as the specification of information variables, which are variables not under the user's control that, if they are not currently known, will be known before the performance of a particular set of decisions.

So extended, inference is performed on junction trees whose topology respects the information order and which has both probability and utility potentials associated with the clusters and intersection sets. Transmission of information through this structure now also includes a utility maximization procedure for each decision variable. The result is decision policies which specify the value to which each (relevant) decision variable ought to be set to maximize expected utility given the evidence that will obtain at the time of the decision. Details of the algorithm can be found in [7].

VIII. A Basic Example

Let us imagine that we are interested in producing a model capable of predicting the state of a set of variables, $\Gamma = \{B\}$, by means of the evidence provided by the set of variables, $\Delta = \{A, C, D, E\}$. To pursue this goal we have a training dataset $\mathfrak{D}$. After learning the network structure and parameters we obtain the network indicated in figure 1 and tables II to VI. (Notice that we use simple probability tables, but in practice we would have Dirichlet distributions. If we were to try and estimate the variance we would need to know the hyper-parameters of the involved Dirichlet distributions.)

-	$A = a_1$	$A = a_2$
-	.6	.4

TABLE II: $P(A)$

A	$B = b_1$	$B = b_2$
a_1	.7	.3
a_2	.4	.6

TABLE III: $P(B|A)$

A	$C = c_1$	$C = c_2$
a_1	.2	.8
a_2	.5	.5

TABLE IV: $P(C|A)$

B	C	$D = d_1$	$D = d_2$
b_1	c_1	.1	.9
b_1	c_2	.2	.8
b_2	c_1	.3	.7
b_2	c_2	.4	.6

TABLE V: $P(D|B, C)$

C	$E = e_1$	$E = e_2$
c_1	.2	.8
c_2	.4	.6

TABLE VI: $P(E|C)$

From the DAG given in figure 1 we see that the Markov Blanket of variable B is $\{A, C, D\}$. Accordingly, if we know we are always able to obtain the value for variable D then we need no longer track variable E.

We are also able to calculate the aposteriori probability distribution of B for any set of evidence of the remaining variables. For example, $P(B|A = a_1, C = c_2, D = d_1) \approx< .54, .46 >$. It is also possible to calculate aposteriori probabilities given soft evidence such as that currently $P(E) =< .96, .04 >$. Were we to estimate the variance we would get confidence intervals around these values according to the density parameter we set. For example, depending on the hyper-parameters of the Dirichlet distribution, we might obtain that 99.5% of the density of the second order probabilities for $P(B = b_1|A = a_1, C = c_2, D = d_1)$ lies between the values .49 and .59.

If we were to specify the utility to use of variable B taking particular values, that variable A was under our control and that we alway know the value of variable D before we set the value of variable A, we can obtain a decision policy given our current knowledge of the system - say that $D = d_1$ - and the expected utility in all cases (see table VII).

IX. A Dynamic Example: Filtering/State Estimation

In the above example, it is natural to assume all variables are current. Note, though, that should the value taken by B depend not only on current values of A,C and D, we may introduce additional dynamic predecessor variables. Each would represent a given variable a given number of discrete time slices previously. We may also seek to find such relationships by including these dynamic predecessors in the domain where performing learning.

A similar, if more wholesale, approach leads to the utilization of Bayesian networks in filtering/dynamic state estimation. For simplicity we will examine a non-decision theoretic case, but the approach we discuss can be extended to permit dynamic decision making.

Imagine that we have a Bayesian network model, B, for a system, S. A 'dynamic Bayesian network', D, of S of degree n will including $n + 1$ copies of the variables of B, such that every variable of B will have its current state and n previous states (utilizing discrete time-slices) represented by a variable in D. If the variables of the system are not directly observable, we include an observation variable set, which permits the entering of current evidence into the network. Likewise, if we can influence the system with current actions, we include a control variable set.

The simplest case is a dynamic network of degree 1, where single variables represent the system being modeled and the system of observations at each time slice. This is a hidden Markov model of the system and an example is given in figure 4. The system transition function is the conditional distribution which relate a previous time slice $t-1$ to the current time slice t. The observation function is the conditional distribution which relate the current state of the system to our observations.

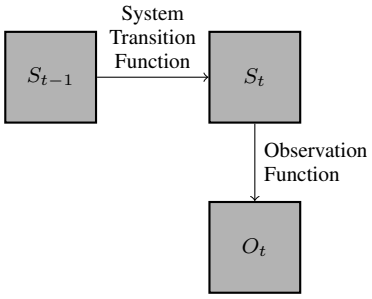

Fig. 4: A Hidden Markov Model

Decomposing the single variable into a network of variables gives us a (more general and more tractable) dynamic Bayesian network. An example is given in figure 5. In such a case, the boxes are now merely conceptual, and the system transition functions and observation functions are represented by the relationships between nodes.

For those familiar with Kalman filters, it should be clear that the Kalman filter is a (dynamic) Bayesian network with linear Gaussian conditional distributions. A discrete Bayesian network permits us to introduce non-linear transition and observation functions in such a way as to perform efficient learning and inference (both when dealing with purely predictive cases and in decision theoretic cases) algorithms. A general Bayesian network is a Bayes filter.

The use of a general Bayes filter faces one additional complication. Since no exact algorithms exist, inference must proceed by sampling. Where the model used is

C	A	$ExpectedUtility$
c_1	a_1	783
c_2	a_2	103

TABLE VII: Sample decision policy

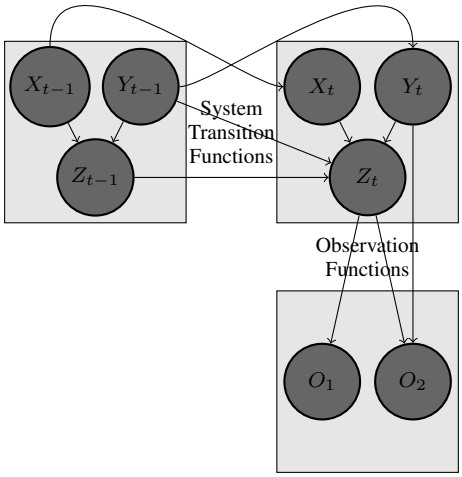

Fig. 5: A Dynamic Bayesian Network that is not a Hidden Markov Model

constructed from causal knowledge, it is common for the topology to be such that, although samples will be weighted by their likelihood given the evidence, the values of the current state variables within samples will be completely independent of the evidence. The effect of this is that to maintain accuracy we must exponentially increase the number of samples in line with t. This problem can be overcome using the particle filtering algorithm, which generates samples at each step by resampling from a set of N samples, weighted by their likelihood given current evidence. The effect is to focus the sampling on high probability regions of the state space. See [8] for a detailed explanation.

Given a set of structures (representing different potential systems/patterns) an extension of the Baum-Welch algorithm (see [9]) permits a likelihood estimation of the most probable structure given a set of observations. We are able to utilize the same algorithm learn the structure and parameters of a 'hidden' dynamic Bayesian network from the observations alone.

It is possible to introduce decision theoretic elements to dynamic Bayesian networks through the use of bounded and unbounded future utility estimations. For unbounded and large bounded cases, tractability concerns currently tend to force the use of sub-optimal estimates.

REFERENCES

[1] S. Monti and G. Cooper, "Technical report: Learning hybrid bayesian networks from data," 1997.

[2] D. Chickering, "Learning equivalence classes of bayesian-network structures," *Journal of Machine Learning Research*, no. 2, pp. 445–498, 2002.

[3] D. Chickering, "Optimal structure identification with greedy search," *Journal of Machine Learning Research*, no. 3, pp. 507–554, 2002.

[4] R. Neopolitan, *Learning Bayesian Networks*. Pearson Prentice Hall, 2004.

[5] C. Huang and A. Darwiche, "Inference in belief networks: A procedural guide," *International Journal of Approximate Reasoning*, no. 11, pp. 1–158, 1994.

[6] T. van Allen, A. Singh, R. Greiner, and P. Hooper, "Quantifying the uncertainty of a belief net response: Bayesian error-bars for belief net inference," *Artificial Intelligence*, no. 172, pp. 483–513, 2008.

[7] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*. Springer, 2nd ed., 2007.

[8] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 3rd ed., 2010.

[9] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Elsevier, 4th ed., 2009.