Probabilistic Reasoning

Michael Rose

In which we explain how to build network models to reason under uncertainty according to the laws of probability theory.

14.1 | Representing Knowledge in an Uncertain Domain

A **Bayesian Network** is a directed graph in which each node is annotated with quantitative probability information. In these networks:

- Each node corresponds to a random variable, which may be discrete or continuous
- A set of directed links or arrows connects each pair of nodes. If there is an arrow from node X to node Y, X is said to be a parent of Y. The graph has no directed cycles, and hence is a directed acyclic graph.
- Each node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node.

The topology of the network specifies the conditional independence relationships that hold in the domain. The combination of topology and the conditional distributions suffices to specify (implicitly) the full joint distribution for all the variables.

14.2 | The Semantics of Bayesian Networks

There are two ways to understand the semantics of a Bayesian network: - The first is to see the network as a representation of the joint probability distribution - The second is to view it as an encoding of a collection of conditional independence statements

These are equivalent, but the first is helpful for constructing networks and the second is helpful in designing inference procedures.

14.2.1 | Representing the Full Joint Distribution

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i))$$

A method for constructing Bayesian Networks

We can rewrite the entries in a joint distribution in terms of conditional probability using the product rule:

$$P(x_1, ..., x_n) = P(x_n|x_{n-1}, ..., x_1)P(x_{n-1}, ..., x_1)$$

Then we repeat the process until we have one big product:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|x_{i-1}, ..., x_1)$$

This is the chain rule, which holds for any set of random variables. Given that these nodes are directed, we can rewrite above as:

$$P(X_i|X_{i-1},...,X_1) = P(X_i|Parents(X_i))$$

provided that $Parents(X_i) \subseteq \{X_{i-1},...,X_1\}$

What this says is that a Bayesian network is a correct representation of the domain only if each node is conditionally independent of its other predecessors in the node ordering, given its parents. We can satisfy this condition with this methodology:

1. Nodes: First determine the set of variables that are required to model the domain. Now order them, $\{X_1,...,X_n\}$. Any order will work, but the resulting network will be more compact if the variables are ordered such that causes precede effects.
2. Links: For $i = 1$ to n, do:
    - Choose from $X_1,...,X_{i-1}$, a minimal set of parents for X_i
    - For each parent insert a link from the parent to X_i
    - Conditional Probability Tables: Write down the conditional probability table, $P(X_i|Parents(X_i))$

Compactness and Node Ordering

As well as being a complete and nonredundant representation of the domain, a Bayesian network can often be far more *compact* than the full joint distribution. The compactness of Bayesian networks is an example of a general property of **locally structured** (also called **sparse**) systems. In a locally structured system, each subcomponent interacts directly with only a bounded number of other components, regardless of the total number of components. Local structure is usually associated with linear rather than exponential growth in complexity.

In Bayesian networks it is reasonable to suppose that in most domains each random variable is directly influenced by at most k others. If we assume n boolean variables for simplicity, then the amount of information needed to specify each conditional probability table will be at most 2^k numbers, and the complete network can be specified by $n2^k$ numbers. In contrast, the joint distribution contains 2^n numbers.

If we stick to a causal model, we end up having to specify fewer numbers, and the numbers will often be easier to come up with.

14.2.2 | Conditional Independence Relations in Bayesian Networks

We have looked at a “numerical” semantic system for Bayesian networks in terms of the representaiton of the full joint distribution. Using this to derive a method for constructing Bayesian networks, by consequence a node is conditionally independent of its other predecessors, given its parents.

We can also go in the other direction - We can start from a “topological” semantic that specifies the conditional independence relationships encoded by the graph structure, and from this we can derive the numerical semantics. The topological semantics specifies that each variable is conditionally independent of its non-descendants, given its parents. In this sense, the numerical and topological semantics are equivalent.

Another important independence property is implied by its topological semantics: a node is conditionally independent of all other nodes in the network, given its parents, children, and children’s parents - that is, given its **Markov Blanket**.

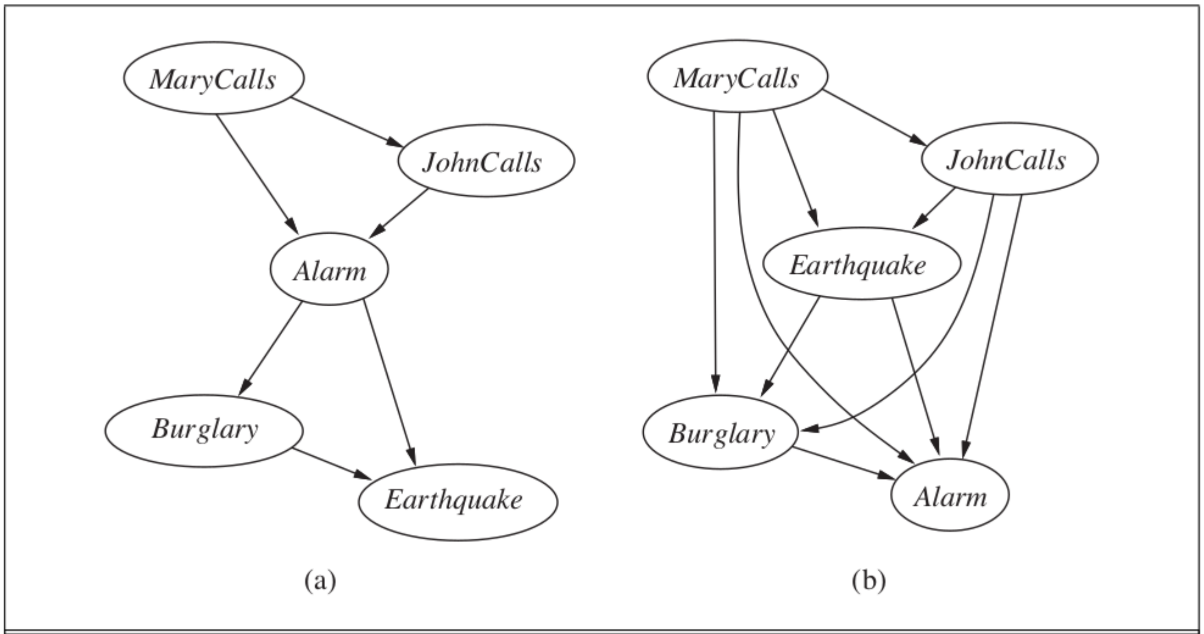

Figure 14.3 Network structure depends on order of introduction. In each network, we have introduced nodes in top-to-bottom order.

Figure 1: Examples of Bayesian Networks

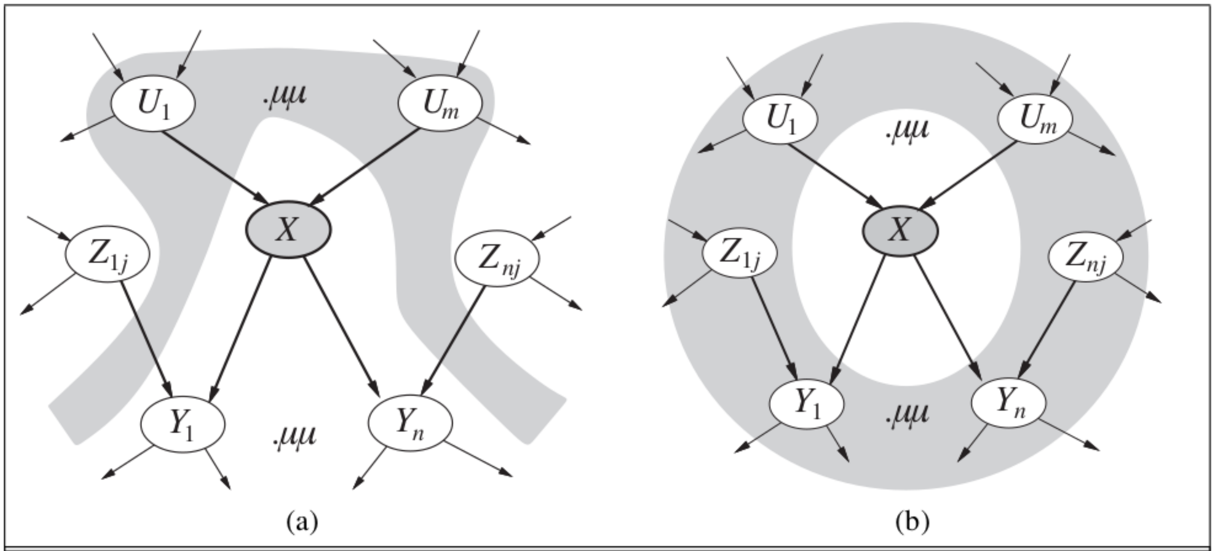

Figure 14.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

Figure 2: Markov Blankets

14.3 | Efficient Representation of Conditional Distributions

Filling in a conditional probability table requires up to $O(2^k)$ numbers. Instead of dealing with thta, usually such relationships are describable by a **canonical distribution** that fits some standard pattern. In such cases, the complete table can be specified by naming the pattern and perhaps supplying a few parameters.

The simplest example is provided by **deterministic nodes**, which have their values specified exactly by the values of their parents, with no uncertainty. For example, if the parent nodes are different prices for a specific make of car at different dealerships and the child is the price that the shopper ends up paying, we can see that the child (may be) the min of the parent nodes - making it determined by the parents.

Uncertain relationships can often be characterized by **noisy** logical relationships. The standard example is the **noisy-OR** relation, which is a generalization of the logical OR. The noisy-OR model allows for uncertainty about the ability of each parent to cause the child to be true - the causal relationship between parent and child may be *inhibited.* For example in propositional logic, Fever may be true iff cold, flu or malaria is true. In the noisy-OR model, a patient could have a cold but not exhibit a fever.

Noisy-OR models make two assumptions: 1. It assumes that all possible causes are listed - if some are missing, we could add a **leak node** which covers miscellaneous causes 2. It assumes that inhibition of each parent is independent of inhibition of any other other parents - For example whatever inhibits Malaria from causing a fever is independent of whatever inhibits Flu from causing a fever

Given these assumptions, Fever is false iff all its true parents are inhibited. In general, noisy logical relationships in which a variable depends on k parents can be described using $O(k)$ parameters instead of $O(2^k)$ for the full conditional probability table.

Bayesian Networks with Continuous Variables

By definition, continuous variables have an infinite number of possible values, so it is impossible to specify conditional probabilities explicitly for each value. One possible way to handle continuous variables is to avoid them by using **discretization** - dividing up the possible values into a fixed set of intervals.

The most common solution for dealing with continuous variables is to define standard families of probability density functions that are specified by a finite number of **parameters** (for example a Gaussian with mean μ and variance σ^2). Another solution is the **nonparametric representation** - which defines the conditional distribution implicitly with a collection of instances, each containing specific values of the parent and child variables.

A network with both discrete and continuous variables is called a **Hybrid Bayesian Network**. To specify a hybrid network, we need to specify two new kinds of distributions: 1. The conditional distribution for a continuous variable given discrete or continuous parents 2. The conditional distribution for a discrete variable given continuous parents.

A common choice for modeling these is the **linear Gaussian distribution**, in which the child has a Gaussian distribution whose mean μ varies linearly with the value of the parent and whose standard deviation σ is fixed. A network containing only continuous variables with linear Gaussian distributions has a joint distribution that is a multivariate Gaussian distribution over all the variables. Furthermore, the posterior distribution given any evidence also has the property.

When discrete variables are added as parents (not as children) of continuous variables, the network defines a **conditional Gaussian distribution**. Given any assignment to the discrete variables, ,the distribution over the continuous variables is a multivariate Gaussian.

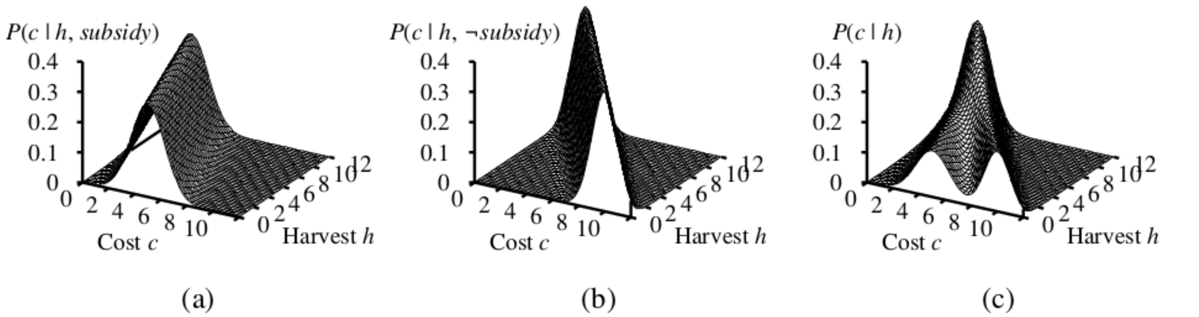

Figure 14.6 The graphs in (a) and (b) show the probability distribution over $Cost$ as a function of $Harvest$ size, with $Subsidy$ true and false, respectively. Graph (c) shows the distribution $P(Cost \mid Harvest)$, obtained by summing over the two subsidy cases.

Figure 3: Hybrid Bayesian Networks

14.4 | Exact Inference in Bayesian Networks

The basic task for any probabilistic inference system is to compute the posterior probability distribution for a set of query variables, given some observed event.

14.4.1 | Inference by Enumeration

Any conditional probability can be computer by summing terms from the full joint distribution.

$$P(X|e) = \alpha P(X, e) = \alpha \sum_y P(X, e, y)$$

A query can be answered using a Bayesian network by computing sums of products of conditional probabilities from the network.

14.4.2 | The variable elimination algorithm

The enumeration algorithm can be improved substantially by eliminating repeated calculations. The idea is to do the calculation once and save the results for later use. This is a form of dynamic programming.

14.4.3 | The Complexity of Exact Inference

The complexity of exact inference in Bayesian networks depends strongly on the structure of the network. The networks be have seen in which arere is at most one undirected path between any two nodes in the network are called **singly connected** networks, or **polytrees**. The time and space complexity of exact inference in polytrees is linear in the size of the network.

For **multiply connected** networks, variable elimination can have exponential time and space complexity in the worst case, even when the number of parent nodes is bounded. Since these networks include inference in propositional logic as a special case, inference in Bayesian networks is NP-hard. In fact, it can be shown

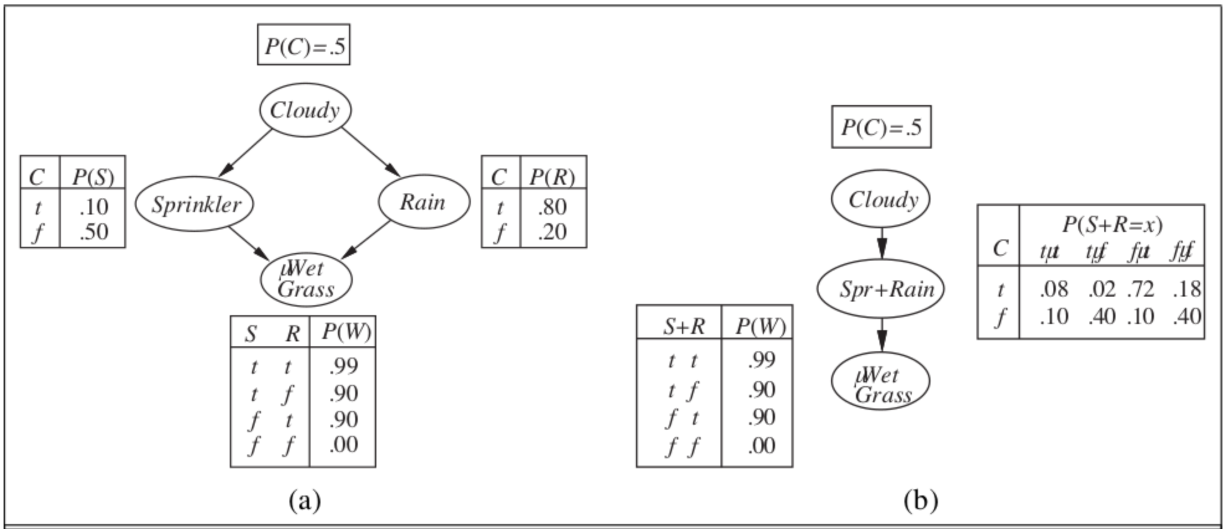

Figure 14.12 (a) A multiply connected network with conditional probability tables. (b) A clustered equivalent of the multiply connected network.

Figure 4:

that the problem is as hard as that of computing the number of satisfying assignments for a propositional logic formula, meaning that is is #P hard - strictly harder than NP-complete problems.

There is a close connection between the complexity of Bayesian network inference and the complexity of constraint satisfaction problems. Measures such as tree width, which bound the complexity of solving a CSP, can also be applied directly to Bayesian networks. Moreover, the variable elimination algorithm can be generalized to solve CSPs as well as Bayesian networks.

14.4.4 | Clustering Algorithms

If we wish to compute posterior probabilities for all the variables in a network, we would need to issue $O(n)$ queries costing $O(n)$ time each, for a total of $O(n^2)$ time. Using **clustering** (or **join tree**) algorithms, this time can be reduced to $O(n)$. Once the network is in polytree form, a special purpose inference algorithm is required. The algorithm is a form of constraint propagation where the constraints ensure that neighboring meganodes agree on the posterior probability of any variables that they have in common.

14.5 | Approximate Inference in Bayesian Networks

This section describes randomized sampling algorithms, also called **Monte Carlo** algorithms.

14.5.1 | Direct Sampling Methods

Let $S_{PS}(x_1, ..., x_n)$ be the probability that a specific event is generated by the prior-sample algorithm. Then, looking at the sampling process we have $S_{PS} = \prod_{i=1}^{n} P(x_i|parents(X_i))$ because each sampling step depends only on the parent values. Then $S_{PS}(x_1, ..., x_n) = P(x_1, ..., x_n)$ since it is exactly the probability of an event according to the Bayesian net's representation of the joint distribution.

The answers are computed by counting the actual samples generated. Suppose there are N tital samples, and let $N_{PS}(x_1, ..., x_n)$ by the number of times that the specific event $(x_1, ..., x_n)$ occurs in the set of samples. We expect this number, as a fraction of the total, to converge in the limit to its expected value according to the sampling probability:

$$\lim_{N \to \infty} \frac{N_{PS}(x_1, ..., x_n)}{N} = S_{PS}(x_1, ..., x_n) = P(x_1, ..., x_n)$$

Rejection Sampling in Bayesian Networks

Rejection Sampling is a general method for producing samples from a hard-to-sample distribution given an easy-to-sample distribution. In its simplest form, it can be used to compute conditional probabilities.

First, it generates samples from the prior distribution specified by the network. Then it rejects all those that do not match the evidence. Finally, the estimate $\hat{P}(X = x|e)$ is obtained by counting how often $X = x$ occurs in the remaining samples.

The biggest problem with rejection sampling is that it drops so many samples. The fraction of samples consistent with the evidence e drops exponentially as the number of evidence variables grows, so the procedure is unusable for complex problems.

Likelihood Weighting

Likelihood Weighting avoids the inefficiency of rejection sampling by generating only events that are consistent with the evidence e. It is a particular instance of the general statistical technique of **importance sampling**, tailored for inference in Bayesian networks.

Likelihood weighting fixes the values for the evidence variables E and samples only the nonevidence variables. This guarantees that each event generated is consistent with the evidence. Before tallying the counts in the distribution for the query table, each event is weighted by the likelihood that the event accords to the evidence, given its parents. Intuitively, events in which the actual evidence appear unlikely are given less weight.

Since likelihood weighting uses all the samples generated, it can be much more efficient than rejection sampling. It will suffer a degradation in performance as the number of evidence variables increases. This is because most samples will have very low weight and hence the weighted estimate will be dominated by the tiny fraction of samples that accord more than an infinitesimal likelihood to the evidence. This problem is exacerbated if the evidence variables occur late in the variable ordering, becayse the nonevidence variables will have no evidence in their parents and ancestors to guide the generation of samples. This means the samples will be simulations that bear little resemblance to the reality suggested by the evidence.

14.5.2 | Inference by Markov Chain Simulation

Markov Chain Monte Carlo algorithms work differently from rejection sampling and likelihood weighting. Instead of generating each sample from scratch, MCMC algorithms generate each sample by making a random change to the preceding sample. It is helpful to think of an MCMC algorithm as being in a particular currect stae specifying a value for every variable and generating a next state by making random changes to the current state.

Gibbs Sampling in Bayesian Networks

The Gibbs sampler for a Bayesian network starts with an arbitrary state (with the evidence variables fixed at their observed states) and generates a next state by randomly sampling a value for one of the nonevidence variables X_i. The sampling for X_i is done conditioned on the current values of the variables in the Markov blanket of X_i. The algorithm wanders randomly around the state space flipping one variable at a time, but keeping the evidence variables fixed.

Essentially, the sampling process settles into a dynamic equilibrium in which the long-run fraction of time spent in each state is exactly proportional to its posterior probability. This property follows from the specific **transition probability** with which the process moves from one state to another, as defined by the conditional distribution given the Markov blanket of the variable being sampled.

Let $q(x \to x')$ be the probability that the process makes a transition from state x to state x'. This transition probability defines the Markov chain on the state space, and let $\pi_t(x)$ be the probability that the system is in state x at time t.

We can calculate $\pi_{t+1}(x')$ by summing, for all the states the system could be in at time t:

$$\pi_{t+1}(x') = \sum_x \pi_t(x) q(x \to x')$$

We say that the chain has reached its **stationary distribution** if $\pi_t = \pi_{t+1}$.

If the transition probability distribution q is **ergodic** then every state is reachable from every other state and there are no strictly periodic cycles - there is exactly one distribution π satisfying this equation for any given q. We can read the equation above as saying that the expected outflow from each state is equal to the expected inflow from all the states.

If the expected inflow between any pair of states is the same in both directions, or

$$\pi(x) q(x \to x') = \pi(x') q(x' \to x) \text{ for all } x, x'$$

then we say that $q(x \to x')$ is in **detailed balance** with $\pi(x)$. It can be shown that detailed balance implies stationarity by summing over x in the equation above.

14.6 | Relational and First-Order Probability Models