Probabilistic Reasoning

Yashovardhan Srivastava

29 June,2021

Introduction

In this chapter, we will learn how to build network models to reason under uncertainty according to the laws of probability theory.This chapter also introduces **Bayesian Networks** to systematically represent probabilistic relationships and provide certainty with uncertain knowledge in a natural and efficient way.

Knowledge Representation in Uncertain Domain

We saw that full joint probability distribution can answer any questions about any domain ,but they grow very large as the number of variables grow large. Also , specifying probabilities for all events (or possible worlds) one by one is tedious.Conditional and independence relationship among variables reduce the numbers of events. **Bayesian networks** are used exactly for this purpose. Bayesian networks can represent any full joint distribution very concisely.

Bayesian Networks

A Bayesian Network is a directed graph in which each node is annotated with the quantitative probability information. The full specification is as follows:

- Each node corresponds to a random variable in the domain(either discrete or continuous).
- A set of directed links connect pair of nodes. An arrow from node **X** to **Y** denotes that **X** is a parent of **Y**.The graph has no directed cycles,and hence is a directed acyclic graph.
- Each node has a conditional probability distribution $\mathbf{P}(X_i|Parents(X_i))$ that quantifies the effect of parents on the node.

The topology of the network specifies the conditional independence relationships among variables. $\mathbf{X} \rightarrow \mathbf{Y}$ denotes that **X** has a *direct influence* on **Y**,which suggests that cause should be the parent of effects(**X** is cause and **Y** is the effect). We will see that the combination of the topology and the conditional distributions suffices to specify the full joint distribution for all the variables.

Probabilities are useful because they summarize a potentially infinite set of circumstances.Most of the details/events(and their infinite list) are basically abstracted under probability For an agent which has to cope with a very large world, Bayesian networks are a very useful tool.

Conditional Probability Table

CPT is a form of table that contains the conditional probability of each node value for a conditioning case[1].Each row must sum to one, as the entries represent exhaustive set of cases for the variable. For a boolean variable with k boolean parents, the CPT contains 2^k independently specifiable probabilities.

[1]A conditioning case is just a possible combination of values for the parent nodes-a miniature possible world

Semantics of Bayesian Network

There are two ways in which we can understand the semantics of Bayesian networks:

- The first way is to see the network as a joint probability distribution.
- The second is to view it as an encoding of a collection of conditional independence statements.

Although the two views are equivalent, the first is useful in understanding *how* to construct the network, while the other is helpful in designing inference procedures.

Bayesian networks as full joint distributions

One way to define the semantics of a Bayesian network is to define the way in which it represents a specific joint distribution over all the variables. A generic entry in the joint distribution is the probability of a conjunction of particular assignments to each variable - such as $P(X_1 = x_1 \wedge x_2 \wedge x_3 ... \wedge X_n = x_n)$. The value of this entry is given by :

$$P(x_1, x_2, ...x_n) = \prod_{i=1}^{n} \theta(x_i | parents(X_i))$$

In this, each entry point in the joint distribution is represented by the product of the appropriate elements of the conditional probability table. From this, we can prove that $\theta(x_i|parents(X_i))$ are exactly the conditional probabilities $\theta(X_i|Parents(X_i))$ implied by the joint distribution.Therefore, we have:

$$P(x_1, x_2, ...x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$

Constructing Bayesian Networks

We will use the equation used above to imply certain conditional independence relations for constructing the network. First, rewrite the entries in the joint distribution using the product rule recursively.

$$P(x_1, x_2, ...x_n) = P(x_n | x_{n-1}, ..., x_1) P(x_{n-1}, ...x_1)$$

Then repeat the process,reducing each conjunctive probability to conditional probability and a smaller conjunction.

$$P(x_1, x_2, ...x_n) = P(x_n | x_{n-1}, ..., x_1) P(x_{n-1} | x_{n-2}, ..., x_1) ... P(x_2|x_1) P(x_1)$$

or

$$P(x_1, x_2, ...x_n) = \prod_{i=1}^{n} P(x_i | x_{i-1}, ..., x_1)$$

Comparing it with the equation we started with, we have:

$$P(x_i | x_{i-1}, ..., x_1) = P(X_i | Parents(X_i))$$

What this means is that the Bayesian network is a correct representation of the domain only if each node is conditionally independent of its predecessors in the node ordering, given its parents.

Compactness and Node Ordering.

Since the Bayesian network are represented as full joint probability distributions, the model is completely determined and can be used for calculating the probability of any proposition.Apart from this, a Bayesian network can be far more compact than the full joint distribution. The compactness of these is due to the general property of locally structured systems - which we derived above.[2]

[2]In locally structured systems, each sub-component interacts directly with only a bounded number of other components,regardless of the total number of components.

Even in a locally structured domain, we will get a compact Bayesian network only if we choose the node ordering well. The model should be designed in **causal** manner, that is from cause to effects, not **diagnostic** model.In making a diagnostic model, we end up specifying additional dependencies between otherwise independent causes.

Bayesian networks as conditional independence relations

Using the semantics defined above, we were able to conclude that a node is conditionally independent of its predecessor, given its parents. Not only that, we can go in the opposite direction from semantics(so-called 'topological' semantics) that specify conditional independent relations and from these we can derive the 'numerical' semantics. The topological semantics specifies that each variable is conditionally independent of its non-descendants, given its parents. From these assertions and the interpretation of network parameters, the full joint distribution can be constructed. In this sense, both 'numerical' and 'topological' semantics are equivalent.

The assertion also highlights an important property - a node is conditional independent of all other nodes in the network given its parents, children, and children"s parents. This is what is known as its **Markov Blanket**.

Exact inference in Bayesian networks

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

Inference by Enumeration

We saw earlier that any conditional probability can be computed by summing terms of full joint distribution. A query can be answered by:

$$\mathbf{P}(X|\mathbf{e}) = \alpha\mathbf{P}(X,\mathbf{e}) = \alpha\sum_{y} P(X,\mathbf{e},\mathbf{y})$$

where $P(x, e, y)$ is the joint probability distribution that can be written as products of conditional probabilities from the network. Therefore, *a query in can be answered using a Bayesian network by computing sums of products of conditional probabilities from the network.*

The semantics of Bayesian networks gives us expression in terms CPT entries, whose inference can be evaluated using ENUMERATION-ASK algorithm, which is similar to backtracking algorithm - which evaluates the expression tree using depth-first recursion.

Approximate inference in Bayesian networks

Given the intractability of exact inference in large,multiply connected networks, it is necessary to consider approximate inference methods for Bayesian network.Some randomized sampling methods(Monte Carlo methods) provide approximate answers for problems whose accuracy depends on the number of samples generated. This section describes two families of algorithms: Direct Sampling and Markov Chain Sampling.

Inference by Direct Sampling Methods

The primitive element in sampling algorithms is the generation of samples from a known probability distribution. The simplest kind of random sampling process for Bayesian networks generates events from a network that has no evidence associated with it. The probability distribution from which the value is sampled is conditioned on the values already assigned to the variable's parents. The algorithm(so-called PRIOR SAMPLE) generates samples from the prior joint distribution specified by the network. Since the sampling answers are calculated by counting the actual samples generated, we expect answer to converge in the limit of its expected value according to the sampling probability, that is the probability of an event can be estimated as the fraction of all complete events generated by the sampling process that match the partially specified event. We discuss two direct sampling algorithms - rejection sampling and likelihood weighting.

Rejection Sampling

Rejection sampling is a general method for producing samples from a hard-to-sample distribution given and easy-to sample distribution. In its simplest form, it can be used to compute conditional probabilities(that is to compute $P(X|\mathbf{e})$). 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 $P(X|\mathbf{e})$ is obtained by counting how many times $X = x$ occurs in the remaining samples.

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

Likelihood Weighting

Likelihood weighting avoids the inefficiency of rejection sampling by generating only that events that are consistent with the evidence. 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 and samples only the non-evidence variables. This guarantees that each event generated is consistent with the evidence then,and since all the events are not equal, each event is weighted by the *likelihood* that event accords to the evidence -as measured by the product of the conditional probabilities for each evidence variable, given its parents.

Because likelihood weighting uses all the samples generated, it can be much more efficient than rejection sampling. It will, however suffer a degradation in the performance as the number of evidence variables increases.

Inference by Monte Carlo Sampling

Markov Chain Monte Carlo(MCMC) algorithms works quite differently than direct sampling methods as instead of generating each example from scratch, MCMC algorithms generate each sample by making a random change to the preceding sample. It is helpful to think of MCMC algorithm as being in a *current state* and generating a *next state* by making random changes to the current state. A particular form of MCMC called Gibbs Sampling is especially well suited for Bayesian networks.

Gibbs Sampling

The Gibbs sampling algorithm for Bayesian networks starts with an arbitrary state and generates a next state by randomly sampling a value for one of the non evidence variables X_i. The samples of 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.

Gibbs sampling returns consistent estimates for the posterior distribution because *the sampling process settles into a "dynamic equilibrium" in which the long run fraction of the time spent in each state is exactly proportional to its posterior probability.*[3]

Relational and First-Order Probability Models

We saw earlier that what representational advantages first-order logic has over propositional logic. First order logic representation are vastly more concise than equivalent propositional descriptions. Bayesian networks are essentially propositional in nature: the set of random variables are fixed and finite, and each has fixed domain of possible values. This fact limits the applicability of Bayesian networks. *If we can find a way to combine probability theory with the expressive power of first-order logic, we expect to be able to increase dramatically the number of problems that can be handled.* The following sections develops a language that allows us to do exactly that(and much more).

[3]The proof is quite difficult to understand (for now).

Possible Worlds

We saw earlier that a probability model defines a set ω of possible worlds with probability $P(\omega)$ for each world ω. For Bayesian networks, the possible worlds are assignments of values to variables(for Boolean case, the possible worlds are similar to those of propositional logic). Similarly, for a first-order probability model, we need the possible worlds to be those of first-order logic. The model also needs to define a probability for each such possible world, just as a Bayesian network defines a probability for each assignment of values to variables. For such models, we can obtain the probability of any first-order logic sentence ϕ as a sum over possible worlds where it is true.(the addition theorem of probability)

$$P(\phi) = \sum_{\omega:\phi} P(\omega)$$

Conditional probabilities $P(\phi|\mathbf{e})$ can be obtained similarly - so in principle we can ask any question we want from our model. The problem, however is that the set of first-order logic models ins infinite. We can deal with this problem by adopting the same technique we used earlier - borrow the semantics not from the standard semantics of first-order logic, but from the database semantics - with the only difference that closed-world assumption is not true,which is obvious because assuming every unknown fact as false doesn't make sense in a probabilistic reasoning system. We can call these models **relational probability models**,or RPMs.

Relational Probability Models

Like first-order logic, RPMs have constants,functions and predicate symbols. For each function if the type of the object is known, many spurious possible worlds are eliminated. It is therefore recommended to assume a **type signature** for each function. The next question to consider is how to do inference in RPMs. One approach is to collect the evidence and the query and the constants therein, construct the equivalent Bayes net,and apply any of the inference methods discussed above(A technique called **unrolling**). The drawback to this is that the Bayes net may be very large. Furthermore, if there are many objects for an unknown relation or function - then some variables might have multiple parents.

The main aim behind the above approach is to unroll RPM to a Bayesian network. This is similar to the method of **propositionalization** for first-order logical inference. The same idea is applied in probabilistic inference.

Open-Universe Probability Models

We used database semantics as a method to avoid the infinite number of first-order logic models as described above. They are appropriate for situations in which we exactly know the set of relevant objects that exist and can identify them unambiguously. However, in many real world settings - object omniscience are simply untenable. Objects never comes with unique IDs attached.

For these reasons, we need to be able to write **open-universe probability models**(OUPM) based on the standard semantics of first-order logic. An OUPM provides a language for writing such models easily while guaranteeing a unique, consistent probability distribution over the infinite space of possible worlds. The basic idea is to borrow the idea of how RPMs manage to define a unique probability distribution over Bayesian networks, but in first-order setting. OUPM allows generative steps(unlike RPMs) that *add objects* to the possible worlds under construction, that is the event being generated is not the assignment of a value but the very *existence* of objects.

Subject to technical conditions of acyclicity and well-foundedness similar to those of RPMs, open universe models define a unique distribution over possible worlds. Inference models for OUPMs exists, but there are some tricky issues involved in designing these algorithms. Research in this area is still at an early stage, but it seems that first-order probabilistic reasoning systems yields a tremendous increase in the effectiveness of AI systems at handling uncertain reasoning.

References

[1] Russel,Stuart and Norvig,Peter.*Artificial Intelligence-A Modern Approach*(3rd ed.).Pearson Education,2010