Gibbs sampling

Transitions probability

The current state. Of this reminds you of simulated annealing from Chapter 4 or WALKS AT from Chapter 7, that is because both are members of the MCMC family.) Here we describe a particular form of MCMC called Gibbs sampling, which is especially well suited for Bayesian networks. (Other forms, some of them significantly more powerful, are discussed in the notes at the end of the chapter.) We will first describe what the algorithm does, then we will explain why it works.

Gibbs sampling in Bayesian networks

The Gibbs sampling algorithm for Bayesian networks starts with an arbitrary state (with the evidence variables fixed at their observed values) 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 $X_i$. (Recall from page 517 that the Markov blanket of a variable consists of its parents, children, and children’s parents.) The algorithm therefore wanders randomly around the state space—the space of possible complete assignments—flipping one variable at a time, but keeping the evidence variables fixed.

Consider the query $P(\text{Rain} | \text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true})$ applied to the network in Figure 14.12(a). The evidence variables $\text{Sprinkler}$ and $\text{WetGrass}$ are fixed to their observed values and the nonevidence variables $\text{Cloudy}$ and $\text{Rain}$ are initialized randomly—let us say to $\text{true}$ and $\text{false}$ respectively. Thus, the initial state is $[\text{true, true, false, true}]$.

Now the nonevidence variables are sampled repeatedly in an arbitrary order. For example:

1. $\text{Cloudy}$ is sampled, given the current values of its Markov blanket variables: in this case, we sample from $P(\text{Cloudy} | \text{Sprinkler} = \text{true}, \text{Rain} = \text{false})$. (Shortly, we will show how to calculate this distribution.) Suppose the result is $\text{Cloudy} = \text{false}$. Then the new current state is $[\text{false, true, false, true}]$.

2. $\text{Rain}$ is sampled, given the current values of its Markov blanket variables: in this case, we sample from $P(\text{Rain} | \text{Cloudy} = \text{false}, \text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true})$. Suppose this yields $\text{Rain} = \text{true}$. The new current state is $[\text{false, true, true, true}]$.

Each state visited during this process is a sample that contributes to the estimate for the query variable $\text{Rain}$. If the process visits 20 states where $\text{Rain}$ is true and 60 states where $\text{Rain}$ is false, then the answer to the query is $\text{NORMALIZE}((20, 60)) = (0.25, 0.75)$. The complete algorithm is shown in Figure 14.16.

Why Gibbs sampling works

We will now show that Gibbs sampling returns consistent estimates for posterior probabilities. The material in this section is quite technical, but the basic claim is straightforward: 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 remarkable property follows from the specific transition probability with which the Markov blanket of the variable being sampled.
Section 14.5. Approximate Inference in Bayesian Networks

function \( G(\text{e}, \text{h}, N) \), returns an estimate of \( P(X) \)

local variables: \( N \), a vector of counts for each value of \( X \), initially zero
\( Z \), the nonevidence variables in \( \text{h} \)
\( x \), the current state of the network, initially copied from \( \text{e} \)

initialize \( x \) with random values for the variables in \( Z \)
for \( j = 1 \) to \( N \) do
  for each \( Z_i \) in \( Z \) do
    set the value of \( Z_i \) in \( x \) by sampling from \( P(Z_i|\text{mb}(Z_i)) \)
  \( N[x] = N[x] + 1 \) where \( x \) is the value of \( X \) in \( x \)
return \( \text{NORMALIZE}(N) \)

Figure 14.16 The Gibbs sampling algorithm for approximate inference in Bayesian networks; this version cycles through the variables, but choosing variables at random also works.

Let \( q(x|x') \) be the probability that the process makes a transition from state \( x \) to state \( x' \). This transition probability defines what is called a Markov chain on the state space. (Markov chains also figure prominently in Chapters 15 and 17.) Now suppose that we run the Markov chain for \( t \) steps, and let \( \pi_t(x) \) be the probability that the system is in state \( x \) at time \( t \). Similarly, let \( \pi_{t+1}(x') \) be the probability of being in state \( x' \) at time \( t+1 \). Given \( \pi_t(x) \), we can calculate \( \pi_{t+1}(x') \) by summing, for all states the system could be in at time \( t \), the probability of being in that state times the probability of making the transition to \( x' \):

\[
\pi_{t+1}(x') = \sum_x \pi_t(x)q(x|x')
\]

We any that the chain has reached its stationary distribution if \( \pi_t = \pi_{t+1} \). Let us cull this stationary distribution \( \pi \); its defining equation is therefore

\[
\pi(x') = \sum_x \pi(x)q(x|x') \quad \text{for all } x'.
\] (14.10)

Provided the transition probability distribution \( q \) is ergodic—that is, every state is reachable from every other and there are no strictly periodic cycles—there is exactly one distribution \( \pi \) satisfying this equation for any given \( q \).

Equation (14.10) can be read as saying that the expected "outflow" from each state (i.e., its current "population") is equal to the expected "inflow" from all the states. One obvious way to satisfy this relationship is if the expected flow between any pair of states is the same in both directions; that is,

\[
\pi(x)q(x|x') = \pi(x')q(x'|x) \quad \text{for all } x, x'.
\] (14.11)

When these equations hold, we say that \( q(x|x') \) is in detailed balance with \( \pi(x) \).

We can show that detailed balance implies stationarity simply by summing over \( x \) in Equation (14.11). We have

\[
\pi(x)q(x|x) = \pi(x')q(x'|x) = \pi(x) \quad \text{for all } x, x'.
\]