CS 188 Introduction to Artificial Intelligence
Spring 2023 Note 18

These lecture notes are based on notes originally written by Josh Hug and Jacky Liang. They have been heavily updated by Regina Wang.

Last updated: January 15, 2023

Approximate Inference in Bayes Nets: Sampling

An alternate approach for probabilistic reasoning is to implicitly calculate the probabilities for our query by simply counting samples. This will not yield the exact solution, as in IBE or Variable Elimination, but this approximate inference is often good enough, especially when taking into account massive savings in computation.

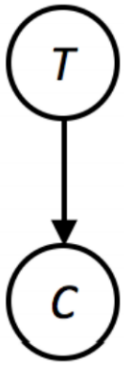
For example, suppose we wanted to calculate $P(+t|+e)$. If we had a magic machine that could generate samples from our distribution, we could collect all samples for which $E = +e$, and then compute the fraction of those samples for while $T = +t$. We'd easily be able to compute any inference we'd want just by looking at the samples. Let's see some different methods for generating samples.

Prior Sampling

Given a Bayes Net model, we can easily write a simulator. For example, consider the CPTs given below for the simplified model with only two variables T and C.

T	P(T)
+t	0.99
-t	0.01

T	C	P(C|T)
+t	+c	0.95
+t	-c	0.05
-t	+c	0.0
-t	-c	1.0



A simple simulator in Python would be written as follows:

```
import random
```

```
def get_t():
    if random.random() < 0.99:
        return True
    return False

def get_c(t):
    if t and random.random() < 0.95:
        return True
    return False

def get_sample():
    t = get_t()
    c = get_c(t)
    return [t, c]
```

We call this simple approach **prior sampling**. The downside of this approach is that it may require the generation of a very large number of samples in order to perform analysis of unlikely scenarios. If we wanted to compute $P(C|-t)$, we'd have to throw away 99% of our samples.

Rejection Sampling

One way to mitigate the previously stated problem is to modify our procedure to early reject any sample inconsistent with our evidence. For example, for the query $P(C|-t)$, we'd avoid generating a value for C unless t is false. This still means we have to throw away most of our samples, but at least the bad samples we generate take less time to create. We call this approach **rejection sampling**.

These two approaches work for the same reason: any valid sample occurs with the same probability as specified in the joint PDF.

Likelihood Weighting

A more exotic approach is **likelihood weighting**, which ensures that we never generate a bad sample. In this approach, we manually set all variables equal to the evidence in our query. For example, if we wanted to compute $P(C|-t)$, we'd simply declare that t is false. The problem here is that this may yield samples that are inconsistent with the correct distribution.

If we simply force some variables to be equal to the evidence, then our samples occur with probability only equal to the products of the CPTs of the non-evidence variables. This means the joint PDF has no guarantee of being correct (though may be for some cases like our two variable Bayes Net). Instead, if we have sampled variables Z_1 through Z_p and fixed evidence variables E_1 through E_m a sample is given by the probability $P(Z_1...Z_p, E_1...E_m) = \prod_i^p P(Z_i|Parents(Z_i))$. What is missing is that the probability of a sample does not include all the probabilities of $P(E_i|Parents(E_i))$, i.e. not every CPT participates.

Likelihood weighting solves this issue by using a weight for each sample, which is the probability of the evidence variables given the sampled variables. That is, instead of counting all samples equally, we can define a weight w_j for sample j that reflects how likely the observed values for the evidence variables are, given the sampled values. In this way, we ensure that every CPT participates. To do this, we iterate through each variable in the Bayes net, as we do for normal sampling), sampling a value if the variable is not an evidence variable, or changing the weight for the sample if the variable is evidence.

For example, suppose we want to calculate $P(T|+c,+e)$. For the jth sample, we'd perform the following algorithm:

- Set w_j to 1.0, and c = true and e = true.
- For T: This is not an evidence variable, so we sample t_j from $P(T)$.
- For C: This is an evidence variable, so we multiply the weight of the sample by $P(+c|t_j)$, i.e. $w_j = w_j \cdot P(+c|t_j)$.
- For S: sample s_j from $P(S|t_j)$.
- For E: multiply the weight of the sample by $P(+e|+c,s_j)$, i.e. $w_j = w_j \cdot P(+e|+c,s_j)$.

Then when we perform the usual counting process, we weight sample j by w_j instead of 1, where $0 <= w_j <= 1$. This approach works because in the final calculations for the probabilities, the weights effectively serve to replace the missing CPTs. In effect, we ensure that the weighted probability of each sample is given by $P(z_1...z_p,e_1...e_m) = \left[\prod_i^p P(z_i|Parents(z_i))\right] \cdot [\prod_i^m P(e_i|Parents(e_i))]$. The pseudocode for Likelihood Weigting is provided below.

function LIKELIHOOD-WEIGHTING(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1,\ldots,X_n)$
    N, the total number of samples to be generated
  local variables: **W**, a vector of weighted counts for each value of X, initially zero

  for j = 1 to N **do**
    x, $w \leftarrow$ WEIGHTED-SAMPLE(bn, **e**)
    $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where x is the value of X in **x**
  return NORMALIZE(**W**)

---

function WEIGHTED-SAMPLE(bn, **e**) **returns** an event and a weight

  $w \leftarrow 1$; **x** $\leftarrow$ an event with n elements initialized from **e**
  foreach variable X_i **in** $X_1,\ldots,X_n$ **do**
    if X_i is an evidence variable with value x_i in **e**
      then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
      else **x**$[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return **x**, w

Figure 14.15 The likelihood-weighting algorithm for inference in Bayesian networks. In WEIGHTED-SAMPLE, each nonevidence variable is sampled according to the conditional distribution given the values already sampled for the variable's parents, while a weight is accumulated based on the likelihood for each evidence variable.

For all three of our sampling methods (prior sampling, rejection sampling, and likelihod weighting), we can get increasing amounts of accuracy by generating additional samples. However, of the three, likelihood weighting is the most computationally efficient, for reasons beyond the scope of this course.

Gibbs Sampling

Gibbs Sampling is a fourth approach for sampling. In this approach, we first set all variables to some totally random value (not taking into account any CPTs). We then repeatedly pick one variable at a time, clear its value, and resample it given the values currently assigned to all other variables.

For the T, C, S, E example above, we might assign t = true, c = true, s = false, and e = true. We then pick one of our four variables to resample, say S, and clear it. We then pick a new variable from the distribution $P(S|+t,+c,+e)$. This requires us knowing this conditional distribution. It turns out that we can easily compute the distribution of any single variable given all other variables. More specifically, $P(S|T,C,E)$ can be calculated only using the CPTs that connect S with its neighbors. Thus, in a typical Bayes Net, where most variables have only a small number of neighbors, we can precompute the conditional distributions for each variable given all of its neighbors in linear time.

We will not prove this, but if we repeat this process enough times, our later samples will eventually converge to the correct distribution even though we may start from a low-probability assignment of values. If you're curious, there are some caveats beyond the scope of the course that you can read about under the Failure Modes section of the Wikipedia article for Gibbs Sampling.

The pseudocode for Gibbs Sampling is provided below.

function GIBBS-ASK(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  local variables: **N**, a vector of counts for each value of X, initially zero
    Z, the nonevidence variables in bn
    x, the current state of the network, initially copied from **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 $\mathbf{P}(Z_i|mb(Z_i))$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in **x**
  return 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.

Conclusion

To summarize, Bayes' Nets is a powerful representation of joint probability distributions. Its topological structure encodes independence and conditional independence relationships, and we can use it to model arbitrary distributions to perform inference and sampling.

In this note, we covered two approaches to probabilistic inference: exact inference and probabilistic inference (sampling). In exact inference, we are guaranteed the exact correct probability, but the amount of computation may be prohibitive.

The exact inference algorithms covered were:

- Inference By Enumeration
- Variable Elimination

We can turn to sampling to approximate solutions while using less compute.

The sampling algorithms covered were:

- Prior Sampling
- Rejection Sampling
- Likelihood Weighting
- Gibbs Sampling