MAC6916 PROBABILISTIC GRAPHICAL MODELS LECTURE 5: SAMPLING ALGORITHMS FOR APPROXIMATE PROBABILISTIC INFERENCE

DENIS D. MAUÁ

1. Introduction

Here and in the next few lectures we study practical algorithms for the *inference problem*, which comes in many forms. To make this discussion concise we will focus on the computation of the probability of evidence $\mathbb{P}(E_1 = e_1, \ldots, E_m = e_m)$ where $E_1, \ldots, E_m$ are the evidence variables and the (partial valuation) $\nu(E_i) = e_i$ is the evidence. This is also known as the *partition function*, as it is equivalent to the computation of the partition function in a (reduced) Markov network. A related problem is *belief updating* or *posterior probability* which consists in computing $\mathbb{P}(X_q = x_q | E_1 = e_1, \ldots, E_m = e_m)$. In principle, we can solve this problem by calling an algorithm that solves the partition problem twice so as to obtain $\mathbb{P}(X_q = x_q, E_1 = e_1, \ldots, E_m = e_m)$ and $\mathbb{P}(E_1 = e_1, \ldots, E_m = e_m)$. While this is actually the approach taken by some algorithms, some techniques employ more efficient ways to obtain the posterior probability directly (and this is particularly true for the algorithms we study here). Nevertheless, the adaptations for producing posterior probabilities are often straightforward and do not affect the generality of the results we will see later.

inference problem

partition function

belief updating

posterior probability

Computing the probability of evidence by enumeration is impractical for all but the smallest models, so that more sophisticated algorithms are needed. In this lecture we analyze *sampling algorithms*, that is, algorithms that operate by generating data sets from the probabilistic models and then using these data sets to generate estimates of the desired probabilities. This family of algorithms is also known as *search-based inference algorithms*, as they perform a search over the space of (partial) valuations of variables, and *simulation algorithms*, as they simulate the phenomenon being modeled. We focus on Monte Carlo approaches which generate samples stochastically (i.e., different runs of the algorithms produce different solutions). The algorithms we examine are *approximate*: their solution is only an approximation of correct value.

sampling algorithms

search-based inference algorithms

simulation algorithms

approximate algorithms

Two criteria are generally used to evaluate approximate algorithms: accuracy and efficiency. The former is a measure of the quality of the solutions produced, and the latter is a measure of the amount of computational resources needed. There is often a trade-off between these two quantities, such that increasing one leads to a decrease in the other.

Date: March 22, 2016.

The most common measures of accuracy are absolute error and relative error. Let p be a (marginal) probability we would like to determine and $\hat{p}$ the output of an approximate algorithm. They are defined as

$$\epsilon_{\text{abs}} = |p - \hat{p}| \qquad \text{[absolute error]}$$
$$\epsilon_{\text{rel}} = \frac{|p - \hat{p}|}{p} \qquad \text{[relative error]}$$

An estimate is an absolute (resp., relative) ϵ-approximation if its absolute (resp., relative) error is at most ϵ. Since the algorithms we will see are stochastic, the accuracy of the algorithm is a random variable. We thus discuss the randomized accuracy, which is the probability $1 - \delta$ of the algorithm producing an ϵ-approximation.

An absolute approximation is useful when the desired number is relatively large and we need only finite precision. For example, if $\mathbb{P}(e) > 1/2$, then an absolute error is usually appropriate. It is less appropriate when $\mathbb{P}(e)$ is small, which is usually the case when m is large. Any relative ϵ-approximation is also an absolute ϵ-approximation. Moreover, relative ϵ-approximations to $\mathbb{P}(E_1 = e_1, \ldots, E_m = e_m)$ and $\mathbb{P}(X_q = x_q, E_1 = e_1, \ldots, E_m = e_m)$ can be combined to produce a relative $\sqrt{\epsilon}$-approximation to $\mathbb{P}(X_Q = x_q | E_1 = e_1, \ldots, E_m = e_m)$, but this is not true for absolute approximations. These arguments suggest that relative error is preferred as an accuracy criteria. As we will see, satisfying a relative error is often more demanding than an absolute error.

Efficiency is typically measured in terms of the running time and memory consumption of the algorithm. The algorithms we will study consume very little memory, and we will often focus on the running time, measured as a function of the number of elementary operations performed (arithmetic operations, list consult or insertion at the end, etc).

2. Monte Carlo Estimation

Suppose we are interested in determining the area $\mu(S)$ of the hashed shape S (i.e., the intersection of the circle and the quadrilateral) in Figure 1, and that we have an efficient way of telling whether any certain point is in the area (e.g., by verifying if it is both inside the circle and the quadrilateral). A popular approximate technique, known as Monte Carlo estimation, is to generate points $x_1, \ldots, x_M$ uniformly within the rectangle and to calculate the desired area as the ratio of points that fell inside and outside the target area. Formally, let B_i be a Boolean random variable such that $B_i = 1$ if $x_i \in S$ and $B_i = 0$ otherwise. The estimate is then given by

$$\hat{\mu}(S) = \frac{1}{M} \sum_{i=1}^{M} B_i \,,$$

When $M \to \infty$ then $\hat{\mu}(S)$ converges in probability to $\mu(S)$. The accuracy of this estimate depends on the number of samples generated, and on the distribution from where points are drawn. For instance, a better estimator is to generate points inside the circle and to approximate $\mu(S)$ be the ratio of points that fell inside and outside the desired area.

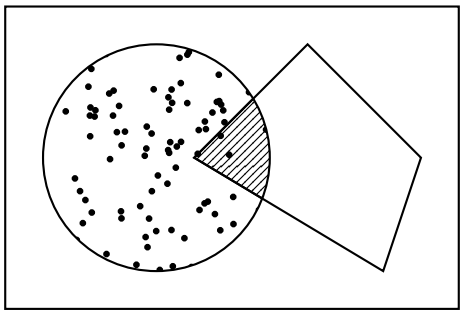

FIGURE 1. Monte Carlo approximation of the length of the hashed area.

3. Logical Sampling

Logical sampling was introduced by [1] and is arguably the simplest non-enumerative algorithm for inference in Bayesian networks. It relies on the assumption that sampling a complete instantiation (valuation) from the joint distribution can be performed efficiently. This is true in discrete Bayesian networks and in many directed models whose local distributions are standard (e.g., Gaussian distributions).[1] Let $X_1, \ldots, X_n$ be a topological ordering of the variables, and consider the evidence $E_1 = e_1, \ldots, E_m = e_m$. A partial valuation is partially defined mapping from variables to values. Any (partial) valuation can be represented as a sequence of values (so that evidence is a partial valuation). We say that two (partial valuations) are *consistent* if they agree on every point where they are both defined. A high-level description of logical sampling is shown below. consitent valuations

(1) Initialize $N \leftarrow 0$
(2) Repeat M times:
    (a) For $i = 1$ to $i = n$ generate $\nu(X_i) = x_i$ with probability $\mathbb{P}(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$
    (b) if $x_1, \ldots, x_n$ is consistent with $e_1, \ldots, e_m$ increase N
(3) return N/M

Note that since the variables are ordered topologically, the distribution $p(X_i|x_1, \ldots, x_{i-1}) = p(X_i|\nu(\mathrm{pa}(X_i)))$. The counter N counts the number of valuations consistent with the evidence, while M is the total number of valuations generated. We thus have that

$$\lim_{M\to\infty} N/M = \mathbb{P}(E_1 = e_1, \ldots, E_m = e_m)\,.$$

Hence, the algorithm asymptotically computes the correct value. What about the finite sample (i.e., finite M) behavior? Let us first analyze the absolute error using the following well-known result.

Theorem 1 (Hoeffding innequality)**.** *Let $B_1, \ldots, B_M$ be a sequence of Bernoulli trials with success probability p, $T = \sum_i B_i/M$ be the average value, and $\mathbb{P}$ be a probability measure over the sequences of Bernoulli trials. Then*

$$\mathbb{P}(T > p + \epsilon) < \exp(-2M\epsilon^2) \quad \textit{and} \quad \mathbb{P}(T < p - \epsilon) < \exp(-2M\epsilon^2)\,,$$

where $\epsilon > 0$.

[1] To learn how to draw a sample from a discrete probability distribution, see Box 12.A on page 489 of [3] and the footnote on page 379 of [2].

absolute error

Hoeffding inequality shows that the *absolute error* (additive error) of the estimator T converges in probability exponentially fast with respect to the number of trials.

Thus, in order to guarantee an ϵ-approximation with probability at least $1-\delta$ with need to set

$$M \geq \ln(2/\delta)/2\epsilon^2 .$$

That upper bound is independent of the desired value, logarithmic in the probability bound δ and polynomial in the absolute error.

The relative error performance requires the following result.

Theorem 2 (Chernoff innequality)**.** *Let $B_1, \ldots, B_M$ be a sequence of Bernoulli trials with success probability p, $T = \sum_i B_i/M$ be the average value, and $\mathbb{P}$ be a probability measure over the sequences of Bernoulli trials. Then*

$$\mathbb{P}(T > p(1+\epsilon)) < \exp(-2Mp\epsilon^2/3) \quad \textit{and} \quad \mathbb{P}(T < p(1-\epsilon)) < \exp(-2Mp\epsilon^2/2) ,$$

where $\epsilon > 0$.

relative error

Chenorff inequality shows that the *relative error* (multiplicative error) of an estimator T converges in probability exponentially fast with respect to the number of trials. This bound however depends on the true probability p (and decays exponentially w.r.t. this value so that for p close to 0 the bound approaches 1).

Using Chernoff inequality, we can bound the number of samples required to get a relative ϵ-approximation estimate with probability at least $1-\delta$ by

$$M \geq 3\ln(2/\delta)/(p\epsilon^2) .$$

The upper bound depends on the unknown target probability p. Hence to be used in practice we must assume some loose upper bound such as $p < 1/2$. When the number of evidence variables is large, this bound is not much informative.

What about the efficiency of the algorithm? An inspection of the code shows that the algorithm takes $O(Mnr\log(d))$ time, where r is the maximum in-degree of a variable, and d is the maximum cardinality of a variable. Hence, the algorithm is polynomial as long as the number of simulations is polynomially bounded. As we have seen, a polynomial bound on the number of samples is insufficient to *guarantee* an accurate estimate when the desired output is small.

The algorithm can be straightforwardly modified to produce several estimates of probabilities with the same set of generated valautions by adding more counters.

4. Importance Sampling

Logical sampling generates samples from the joint distribution $p(\mathcal{V})$ in order to estimate a marginal probability $p(e_1, \ldots, e_m)$. An alternative is to sample points directly from $p(\mathcal{V} - \{E_1, \ldots, E_m\} | e_1, \ldots, e_m)$, thus reducing the space and improving the estimates.

Let $\mathcal{X} = \mathcal{V} - \{E_1, \ldots, E_m\}$. Generate (partial) valuations $\nu_1, \ldots, \nu_M$ from a proposal distribution *proposal distribution* $q(\mathcal{X})$ and estimate the probability as

$$\frac{1}{M}\sum_{i=1}^{M} W_i , \quad \text{where } W_i = p(\nu_i)/q(\nu_i) .$$

The proposal distribution is assumed to satisfy $q(\nu) > 0$ if $p(\nu) > 0$. This ensures that every valuation with positive probability is eventually generated. When this

is true, the asymptotic value of the estimate is

$$\lim_{M\to\infty} \frac{1}{M}\sum_i W_i = \sum_{\nu:\nu(E_i)=e_i} \frac{p(\nu)}{q(\nu)} q(\nu) = p(e_1,\ldots,e_m)\,.$$

When the proposal distribution is chosen carefully, importance sampling produces more reliable estimates (in the sense that they are consistent and have smaller variance). Choosing good proposal distributions is however not always easy. One particularly good proposal is used in *likelihood weighting*, where we let $q(\mathcal{X})$ be the joint distribution of the Bayesian Network obtained by re-defining $p(E_i = e_i|\mathrm{pa}(E_i)) = 1$ and $p(E_i \neq e_i|\mathrm{pa}(E_i)) = 0$ for every evidence variable and leaving the remaining local models unchanged. One can verify that

likelihood weighting

$$W_i = \prod_i p(e_i|\nu(\mathrm{pa}(E_i)))\,.$$

Moreover, $q(\nu) \geq p(\nu)$, which makes likelihood weighting asymptotically consistent and with smaller variance (which implies faster convergence).

While likelihood weighting often improves on logical sampling (especially for conditional probabilities), their convergence rate are similar and might require exponentially many valuations to guarantee a certain accuracy. The *bounded-variance algorithm* [4] uses a stopping rule to dynamically decide the number of valuations based on the current estimate and is guaranteed to be polynomial when the number of evidence variables is small. The algorithm assumes that $0 < \ell_i \leq \mathbb{P}(E_i = e_i|\mathrm{pa}(E_i)) \leq u_i < 1$ and proceeds as follows. Once more, assume $X_1,\ldots,X_n$ are ordered topologically.

bounded-variance algorithm

(1) Define $U = \prod_i u_i$, $N^* = 4\ln(2/\delta)(1+\epsilon)/\epsilon^2$
(2) Initialize $N \leftarrow 0, M \leftarrow 0$
(3) While $N < N^*$ repeat:
    (a) $W \leftarrow 1$
    (b) For $i = 1$ to $i = n$ generate $\nu(X_i) = x_i$ with probability $\mathbb{P}(X_i = x_i|X_1 = x_1,\ldots,X_{i-1} = x_{i-1})$ if $X_i \neq E_j$ else do $W \leftarrow W \cdot \mathbb{P}(X_i = e_j|x_1,\ldots,x_{i-1})$
    (c) Do $N \leftarrow N + W/U$ and $M \leftarrow M + 1$
(4) return $U \cdot N/M$

The algorithm is similar in spirit to likelihood weighting, but uses a different estimator that divides by an upper bound in order to reduce (and even bound) the variance.

So far we have discussed algorithms for inference in Bayesian networks, where we can generate valuations directly from the model. This is no longer true in Markov networks. Assume that we want to compute the partition function of a distribution $p(\mathcal{V}) = \phi(\mathcal{V})/Z$ as in Markov networks. We generate valuations $\nu_1,\ldots,\nu_M$ from some proposal distribution $q(\mathcal{V})$, and compute

$$\tilde{Z} = \frac{1}{M}\sum_{i=1}^{M} \tilde{W}_i\,, \quad \text{where } \tilde{W}_i = \phi(\nu_i)/q(\nu_i)\,.$$

In the limit, we have that

$$\lim_{M\to\infty} \frac{1}{M}\sum_i \tilde{W}_i = \sum_\nu \frac{\phi(\nu)}{q(\nu)} q(\nu) = Z\,.$$

Hence, the partition function can be computed much like a probability of evidence (and vice-versa). Any probability can now be computed by importance sampling with weights $[\phi(\nu)/Z]/q(\nu)$. It remains to decide how to choose a proposal distribution. One possibility is to fix an arbitrary topological ordering among the variables and locally normalize every clique potential to turn the Markov network into a Bayesian network with local models $p(X_i|\text{pa}(X_i)) \propto \phi(\mathcal{C}_j)/p(\text{pa}(X_i))$, where $\mathcal{C}_j$ is a clique containing X_i belongs. This will often require chordalizing the network. A usually more successful alternative is Gibbs sampling, which we discuss next.

5. Gibbs Sampling

Gibbs sampling is a general scheme for sampling from complex distributions where generating samples from local conditional models is easy.[2] The algorithm can be succinctly described as follows. Consider any ordering of the variables $X_1, \ldots, X_n$.

(1) Initialize ν_0 with $\nu_0(E_i) = e_i$ and otherwise arbitrary, set $N \leftarrow 0$
(2) Repeat for a number of steps M:
    (a) For $i = 1$ to $i = n$ generate $\nu_j(X_i) = x_i$ with probability $\mathbb{P}(X_i = x_i | X_1 = \nu_j(X_1), \ldots, X_{i-1} = \nu_j(X_{i-1}), X_{i+1} = \nu_{j-1}(X_{i+1}), \ldots, X_n = \nu_{j-1}(X_n))$
    (b) If ν_j is consistent with $e_1, \ldots, e_m$ increment N
    (c) Increment j
(3) Return N/M

Note that a value $X_i = x_i$ is generated given a full valuation. Hence, the probability $\mathbb{P}(X_i = x_i | X_1 = \nu_{j-1}(X_1), \ldots, X_{i-1} = \nu_{j-1}(X_{i-1}), X_{i+1} = \nu_{j-1}(X_{i+1}), \ldots, X_n = \nu_{j-1}(X_n))$ is taken from the conditional probability of X_i given its Markov blanket. One can show that $\mathbb{P}(X_i|\text{ne}(X_i))$ can be efficiently computed in Markov networks (note that $\text{ne}(X_i)$ is the Markov blanket of X_i) and used to generate values x_i. The estimator N/M is known as the *histogram estimator*.

histogram estimator

Under some mild conditions, as $M \to \infty$, the valuations ν_j are generated according to the joint distribution $p(\mathcal{V})$ of the Bayesian network or Markov network. Hence, we can use Gibbs sampling to perform many sorts of inference problems (computing the partition function, computing all marginals, computing an expectation, etc.). When computing conditional probabilities, we can fix the valuations at the evidence; Gibbs sampling is then guaranteed asymptotically to generate valuations from $p(X_q|e_1, \ldots, e_m)$. In this case a theoretically improved estimator is a theoretically improved alternative estimator is the *mixture estimator*:

mixture estimator

$$\hat{p}(x_i|e_1, \ldots, e_m) = \frac{1}{M} \sum_{j=1}^{M} \mathbb{P}(X_i = \nu_j(X_i)|\nu_j(\text{pa}(X_i)))\,,$$

Analyzing the finite-sample convergence of Gibbs sampling is difficult, and generally there are no non-trivial upper bounds on the necessary number of iterations necessary to guarantee good performance. There are many convergence tests that can be made to verify *estimate* convergence. For example, a simple approach is to run many independent simulations concurrently (using different initializations) and to decide convergence when the estimates produced by the different runs become similar. This is usually accomplished by comparing the variance of each estimate

[2]The theoretical underpinnings of the algorithm are out of the scope of this course.

with respect to the overall estimate (the one using valuations from all runs). Upon convergence, this variance should be small (see Box 12.B on [3, page 522]).

The consistency of the samples generated by Gibbs sampling are only valid asymptotically. For small sample sizes, consecutive valuations are *correlated*, which can greatly bias the estimates produced. To decrease correlation between valuations, practitioners often adopt a *burn-in* phase, during which the generated valuations are discarded. It is customary to have a burn-in sample size of thousands of valuations (these do not contribute to the value of N or M). burn-in

Gibbs sampling has smaller theoretical guarantees than logical and importance sampling. On the other hand, its convergence is not generally affected by the probability being sought, which makes it particularly competitive when computing the probability of rare events. There are also many improvements (e.g., block and Rao-Blackwellized Gibbs sampling) that can speed up convergence of the algorithm.

All methods presented here can be used with continuous variables (as long samples from the local distributions can be efficiently generated) with little additional effort. Sampling methods are usually the choice when devising new complex models (e.g., specialized topic models and probabilistic logics).

6. Reading

There is no recommended reading this week. For more information about sampling algorithms, check out chapter 15 of [2] and chapter 12 of [3].

7. Exercises

Exercise 1. *Let (G, p) be a Markov network factorizing as $p(\mathcal{V}) = \prod_j \phi_j(\mathcal{C}_j)/Z$. Show that*

$$p(X|ne(X)) = \frac{\prod_{j:X\in\mathcal{C}_j} \phi_j(\mathcal{C}_j)}{\sum_X \prod_{j:X\in\mathcal{C}_j} \phi_j(\mathcal{C}_j)}$$

for any variable $X \in \mathcal{V}$.

Exercise 2. *Obtain a similar expression for $p(X|MB(X))$ in the case of Bayesian networks, where $MB(X)$ denotes the Markov Blanket of X (i.e., parents, spouses and children).*

8. Assignment

There is no practical assignment this week. The algorithms discussed in here will be part of a future assignment. Yet, you should start implementing and testing the following parts.

- Implement logical sampling, likelihood weighting and Gibbs sampling.
- Analyze each algorithm. Consider aspects such as accuracy (absolute and relative errors), convergence (accuracy vs. number of samples), efficiency (accuracy vs. runtime), effect of evidence on accuracy and runtime, effect of model size (number of variables) and query size (number of evidence variables) on accuracy, and effect of probability values (input and output) on accuracy and runtime. Write report describing implementation and empirical results.

Use the Bayesian networks provided in the website (in UAI format).

References

[1] M. Henrion. Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling In *Proceedings of the 2nd Conference on Uncertainty in Artificial Intelligence*, pp. 149–163, 1986.

[2] A. Darwiche. Modeling and reasoning with Bayesian networks. Cambridge University Press, 2009.

[3] D. Koller and N. Friedman. Probabilistic Graphical Models MIT Press, 2009.

[4] P. Dagum and M. Luby An optimal approximation algorithm for Bayesian inference. *Artificial Intelligence*, volume 93, issue 1–2, pp. 1–27, 1997.

[5] P. Dagum and M. Luby Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, volume 60, issue 1, pp. 141–153, 1993.