Approximate Inference in Bayesian Networks

Luke Runnels

September 14, 2023

1 Introduction

1.1 Inference in Bayesian Networks

Bayesian networks are some of the most powerful data structures in the field of Artificial Intelligence. These structures allow computer scientists to model a series of random variables by encapsulating their uncertainty and conditional independencies into a single graphical representation.

One of the most significant tasks that one can perform on a bayesian network is inference. Inference is the task of answering queries about a certain random variable in the network, given a set of fixed evidence on other random variables in the network. Traditional exact inference methods calculate these queries by summing over the joint probability distributions from the query, evidence, and hidden random variables that are involved in a inference query. However, this approach is completely unfeasible with large and complex structures, as the time complexity of exact inference can reach into the realm of $O(E^N)$, where E can be thought of as the average number of parents in each random variable and N is the number of random variables in the network.

To get around the problem of exponential time complexity, computer scientists have restored to approximate inference methods using Monte Carlo algorithms. These algorithms essentially sample from the random variables based on the conditional probability table distributions embedded within the network. Although there are different approaches to sampling, the most useful techniques involve sampling through correction weighting and sampling from previous states, which ensures that all samples will be useful in some capacity.

2 Dataset used

2.1 Bayesian network dataset

To evaluate the effectiveness and scalability of the following algorithms, a dataset of polytree bayesian networks was conceived. This dataset consists of three units, where each unit has a set of three polytree bayesian networks. The three bayesian networks consist of five, fifteen, and twenty-five random variables respectively.

In this paper, the relative effectiveness of choosing evidence variables that are upstream and downstream from the query variable, as well as choosing probability entries that are close to and distant from 0 and 1 are evaluated.

To assess the effectiveness of using close and distant CPT entries from 0 and 1, three sets of the three polytree networks were generated using the beta distribution. Using $\alpha = 1$ and $\beta = 1$, $\alpha = 0.5$ and $\beta = 1$, and $\alpha = 1$ and $\beta = 0.5$ as hyperparameters, polytree networks with CPT entries that are uniformly distributed between 0 and 1, trending to 0, and trending to 1 were generated. In summarize, the dataset consists of a total of nine networks, with three 5-polytrees, three 15-polytrees, and three 25-polytrees. One set of the three polytrees has CPT entries uniformly distributed, while the other two sets of polytrees has CPT entries trending towards 0 and 1.

To assess the use of upstream and downstream evidence variables, the query variable for all of the nine networks was placed in the middle according to the network's topological order. For the 5-polytree networks, the first two and last two variables served as evidence. For 15-polytree networks, the first four and last four variables served as evidence. For 25-polytree networks, the first six and last six variables served as evidence. Just like with placing the query variable, the evidence variables were choosen according to the topological order of each of the networks.

On a final note, the implementation for generating this dataset relied on code developed by Chad Crawford in the TU masters group. [1]

3 Algorithm Explanations

3.1 Likelihood Weighting

Likelihood weighting works by making numerous samples from a bayesian network while keeping track of a weighted count for all of the evidence variables. Based on figure 13.18 from Russell el at. [3], to approximate the probability distribution W for a query variable Q, a sample $\mathbf{x}$ is initially fixed by all of the evidence variables in $\mathbf{E}$, with a separate local variable $\mathbf{w}$ that keeps track of the weights. Then, for every random variable in the network, $\mathbf{x}$ and $\mathbf{w}$ are updated via

$$Sample(RV) = \begin{cases} \mathbf{w} = \mathbf{w} * P(x|parents(x)) & \text{if RV is in } \mathbf{E} \\ \mathbf{x}[i] = \text{random sample from } \mathbf{P}(X_i|parents(X_i)) & \text{if RV is not in } \mathbf{E} \end{cases} \tag{1}$$

where i is the current random variable being sampled. The probability distribution W for Q is updated by

$$W[i] = W[i] + w \tag{2}$$

where *W[i]* is the value of Q sampled from $\mathbf{x}$. The equations (1) and (2) consist of a single sample routine for this method.

For many situations, this is an effective sampling method. However, as discussed by Russell el at. [3], the performance of this method decreases if there are a large number of evidence variables, and the evidence variables appear "downstream" from the query variable. To address these problems, variations of Markov Chain Monte Carlo algorithms are used instead.

3.2 Markov Chain Monte Carlo Algorithms

Markov Chain Monte Carlo Algorithms function differently from direct sampling methods such as likelihood weighting. Based on the discussion from Russell el at. [3], these algorithms will sample values in order to maintain a state in the network. This allows for the algorithms make wandering samplings around the network, which as discussed previously, can help alleviate the inefficiencies with downstream evidence and large amounts of evidence variables.

3.2.1 Markov Blankets

To effectively make wandering samples from the network, Monte Carlo Algorithms rely on sampling a random variable from a network given its markov blanket. A markov blanket of a variable is simply the parents, the children, and any other parents of the children. As shown by Russell el at. [3], the probability distribution of a variable given its markov blanket can be calculated as follows.

$$\mathbf{P}(X_i|mb(X_i)) = \alpha P(x_i|parents(X_i)) \prod_{Y_j \in Children(X_i)} P(y_j|parents(Y_j)) \tag{3}$$

where α is a standard normalization constant, x_i refers to every possible value of X_i, and y_j refers to every possible value of Y_j.

3.2.2 Gibbs Sampling

Gibbs sampling works by initially fixing a state sample $\mathbf{x}$ based on the evidence variables $\mathbf{E}$ and count distribution $\mathbf{C}$ for the query X. Then, for the number of samples that are being requested, a non-evidence variable Z from the network is chosen to be sampled from based on the markov blanket distribution shown in (3). Once this sampling has taken place, $\mathbf{x}$ is updated based on the sampled value from Z. Finally, $\mathbf{C}$ is incrementally updated based on

the value of X that is stored in $\mathbf{x}$.

The normalization of $\mathbf{C}$ is denoted as the estimated distribution of X.

3.2.3 Metropolis Hasting Sampling

Metropolis Hasting sampling can be thought of as the general case for gibbs sampling. To maintain a global sample state $\mathbf{Z}$, instead of accepting sample states from Gibbs sampling all of the time, we define a proposal distribution $q(\mathbf{x}'|\mathbf{x})$ for accepting or rejecting a sample generated by gibbs sampling.

Based on suggestions from Russell el at. [3], the proposal distribution is implemented as generating a sample state from 20 Gibbs sampling iterations for a given a probability p. Otherwise, an entirely new sample state is generated using equation (1) from Likelihood Weighting.

The sample state from Gibbs sampling will always be accepted. However, to accept the sample state generated from equation (1), the weight, which is calculated in equation (1) along with the sample state, from the generated sample needs to be greater than the associated weight the current sample state.

Similarly to Gibbs sampling, the distribution for the query variable X will be maintained by a count distribution $\mathbf{C}$, where it is incrementally updated using the value of x_i from $\mathbf{Z}$ after each sampling from $q(\mathbf{x}'|\mathbf{x})$.

4 Performance & Analysis

In the following sections, all of the algorithms described in section 3 will be tested. Each subsection will test these algorithms on a set of polytrees based on their beta distribution, which tests for CPT uniformity, trending towards zero, or trending towards one. For the polytree sizes of 5, 15, and 25 in each sub-subsection, the mean and standard deviation of the PTrue probability, PFalse probability, and runtime in seconds will be reported from 25 trial runs. There will be two sections in a table to measure if the inference algorithms were tested using upstream evidence or downstream evidence variable as specified in the dataset from section 2.

In addition to the algorithms from section 3, the exact inference algorithm's results will be reported at the top of each table. These results will serve as a baseline for evaluating the performance.

It should be noted that Likelihood weighting and Gibbs sampling are being tested in accordance with their descriptions in section 3. However, for better performance, Metropolis Hasting will sample from a Gibbs sampling sub-routine 100 times per sample, instead of the 20 times recommended by Stuart el at. [3]. Also, there will be three instances of testing in Metropolis Hasting, with a single instances having the probability of 0.75, 0.85, and 0.95 for

choosing Gibbs sampling per sample.

On a final note, the exact inference method for comparison is Enumeration Ask. The implementation used here was developed by Robert Geraughty in the TU Masters group [2].

4.1 Performance of Polytrees with Uniform Distribution ($\alpha = 1$ and $\beta = 1$)

Polytree with 5 nodes: Uniform Distribution							
		Upsteam Evidence			Downstream Evidence		
Inference	Measure	PTrue	PFalse	Runtime	PTrue	PFalse	Runtime
Enumeration Ask	Mean	0.28374	0.71625	1.94367	0.30451	0.69548	1.98302
	STD	0	0	0	0	0	0
Likelihood Weighting	Mean	0.28767	0.71232	0.05288	0.30394	0.69605	0.05267
	STD	0.01604	0.016041	0.00549	0.01331	0.01331	0.003912
Gibbs Sampling	Mean	0.28744	0.71256	0.09733	0.30768	0.69232	0.09874
	STD	0.03044	0.03044	0.00953	0.03654	0.036547	0.00551
Metropolis Hasting(0.75)	Mean	0.29198	0.70801	17.73562	0.30741	0.69258	19.56687
	STD	0.01875	0.01875	0.25674	0.01661	0.01661	0.53095
Metropolis Hasting(0.85)	Mean	0.28023	0.71976	21.07437	0.29974	0.70025	22.23812
	STD	0.01941	0.48827	0.48827	0.020339	0.02033	0.40201
Metropolis Hasting(0.95)	Mean	0.27892	0.72107	23.47437	0.30513	0.69486	24.58375
	STD	0.01355	0.01355	0.24717	0.01641	0.01641	0.36679

Polytree with 15 nodes: Uniform Distribution							
		Upsteam Evidence			Downstream Evidence		
Inference	Measure	PTrue	PFalse	Runtime	PTrue	PFalse	Runtime
Enumeration Ask	Mean	0.37677	0.62322	1.97672	0.15114	0.84885	2.09805
	STD	0	0	0	0	0	0
Likelihood Weighting	Mean	0.37296	0.62703	0.12503	0.15552	0.84447	0.12362
	STD	0.01419	0.01419	0.00861	0.01411	0.01411	0.00461
Gibbs Sampling	Mean	0.38036	0.61964	0.21871	0.14064	0.85935	0.22190
	STD	0.08869	0.08869	0.01105	0.05398	0.05397	0.00487
Metropolis Hasting(0.75)	Mean	0.37238	0.62761	42.42375	0.16499	0.83500	44.30687
	STD	0.02141	0.02141	0.76721	0.01329	0.01328	1.07768
Metropolis Hasting(0.85)	Mean	0.37586	0.62413	48.37437	0.15840	0.84159	49.78437
	STD	0.01548	0.01547	0.97470	0.01487	0.01487	0.81388
Metropolis Hasting(0.95)	Mean	0.37966	0.62033	53.95937	0.15664	0.84335	55.7825
	STD	0.01424	0.01424	0.68782	0.00853	0.00853	0.64483

Polytree with 25 nodes: Uniform Distribution							
		Upsteam Evidence			Downstream Evidence		
Inference	Measure	PTrue	PFalse	Runtime	PTrue	PFalse	Runtime
Enumeration Ask	Mean	0.46052	0.53948	22.65639	0.45701	0.54298	86.86238
	STD	0	0	0	0	0	0
Likelihood Weighting	Mean	0.46556	0.53444	0.20066	0.45699	0.54300	0.19698
	STD	0.01287	0.01287	0.01326	0.01914	0.01914	0.00885
Gibbs Sampling	Mean	0.42988	0.57012	0.34017	0.43603	0.56396	0.33991
	STD	0.09264	0.09264	0.00687	0.10999	0.10999	0.00799
Metropolis Hasting(0.75)	Mean	0.45158	0.54841	66.10375	0.44859	0.55141	66.93437
	STD	0.02300	0.02300	1.47303	0.01703	0.01703	1.38945
Metropolis Hasting(0.85)	Mean	0.44683	0.55316	75.58000	0.44143	0.55856	76.55437
	STD	0.01655	0.01655	0.97929	0.01885	0.01885	0.98223
Metropolis Hasting(0.95)	Mean	0.45111	0.54889	84.5875	0.44563	0.55436	85.4775
	STD	0.01884	0.01884	0.86540	0.01682	0.01682	0.94787

4.2 Performance of Polytrees with probability distribution trending towards zero ($\alpha = 0.5$ and $\beta = 1$)

Polytree with 5 nodes: $\alpha = 0.5$ and $\beta = 1$							
		Upsteam Evidence			Downstream Evidence		
Inference	Measure	PTrue	PFalse	Runtime	PTrue	PFalse	Runtime
Enumeration Ask	Mean	0.87613	0.12386	1.96465	0.98594	0.01405	1.92416
	STD	0	0	0	0	0	0
Likelihood Weighting	Mean	0.87788	0.12211	0.05656	0.98584	0.01415	0.05608
	STD	0.01058	0.01058	0.00809	0.00143	0.00143	0.00764
Gibbs Sampling	Mean	0.82307	0.17692	0.09912	0.98324	0.01676	0.10377
	STD	0.11782	0.11782	0.00428	0.00843	0.00843	0.00361
Metropolis Hasting(0.75)	Mean	0.85198	0.14801	19.03062	0.98901	0.01098	20.19312
	STD	0.01554	0.01554	0.53716	0.00284	0.00284	0.54017
Metropolis Hasting(0.85)	Mean	0.84487	0.15512	21.65687	0.98805	0.01194	22.64
	STD	0.01171	0.01171	0.42986	0.00341	0.00341	0.35238
Metropolis Hasting(0.95)	Mean	0.84303	0.15696	24.24687	0.98533	0.01466	25.43625
	STD	0.01455	0.01455	0.55561	0.00396	0.00396	0.55561

Polytree with 15 nodes: $\alpha = 0.5$ and $\beta = 1$							
		Upsteam Evidence			Downstream Evidence		
Inference	Measure	PTrue	PFalse	Runtime	PTrue	PFalse	Runtime
Enumeration Ask	Mean	0.01394	0.98605	2.02184	0.45522	0.54477	2.22861
	STD	0	0	0	0	0	0
Likelihood Weighting	Mean	0.01344	0.98656	0.12818	0.45170	0.54830	0.12638
	STD	0.00371	0.00371	0.01366	0.01548	0.01548	0.00706
Gibbs Sampling	Mean	0.01620	0.98380	0.21840	0.42684	0.57316	0.22360
	STD	0.01649	0.01649	0.00462	0.18330	0.18330	0.00541
Metropolis Hasting(0.75)	Mean	0.01399	0.98601	42.67000	0.36695	0.63305	44.09500
	STD	0.00577	0.00577	1.08354	0.01588	0.01588	0.91049
Metropolis Hasting(0.85)	Mean	0.01335	0.98665	48.50062	0.36867	0.63133	49.28375
	STD	0.00396	0.00396	0.91929	0.01777	0.01777	0.79512
Metropolis Hasting(0.95)	Mean	0.01259	0.98741	54.35813	0.35181	0.64819	55.35938
	STD	0.00435	0.00435	0.64825	0.01738	0.01738	0.75508

Polytree with 25 nodes: $\alpha = 0.5$ and $\beta = 1$							
		Upsteam Evidence			Downstream Evidence		
Inference	Measure	PTrue	PFalse	Runtime	PTrue	PFalse	Runtime
Enumeration Ask	Mean	0.01726	0.98273	26.71657	0.04149	0.95850	26.71657
	STD	0	0	0	0	0	0
Likelihood Weighting	Mean	0.01788	0.98212	0.19504	0.04144	0.95856	0.19399
	STD	0.00503	0.00503	0.00980	0.00711	0.00711	0.00653
Gibbs Sampling	Mean	0.01932	0.98068	0.33815	0.04744	0.95256	0.34210
	STD	0.04283	0.04283	0.00574	0.09346	0.09346	0.00353
Metropolis Hasting(0.75)	Mean	0.01431	0.98569	66.79625	0.02274	0.97726	67.56875
	STD	0.00396	0.00396	1.49560	0.00534	0.00534	1.28942
Metropolis Hasting(0.85)	Mean	0.01371	0.98629	75.35125	0.02330	0.97670	76.33750
	STD	0.00551	0.00551	1.08312	0.00550	0.00550	1.16195
Metropolis Hasting(0.95)	Mean	0.01383	0.98617	85.10313	0.02665	0.97335	85.32250
	STD	0.00249	0.00249	1.09112	0.00539	0.00539	1.10290

4.3 Performance of Polytrees with probability distribution trending towards one ($\alpha = 1$ and $\beta = 0.5$)

Polytree with 5 nodes: $\alpha = 1$ and $\beta = 0.5$							
		Upsteam Evidence			Downstream Evidence		
Inference	Measure	PTrue	PFalse	Runtime	PTrue	PFalse	Runtime
Enumeration Ask	Mean	0.49035	0.50964	1.98631	0.72384	0.27615	1.92018
	STD	0	0	0	0	0	0
Likelihood Weighting	Mean	0.49088	0.50912	0.05659	0.72428	0.27572	0.05724
	STD	0.01206	0.01206	0.00700	0.01517	0.01517	0.00721
Gibbs Sampling	Mean	0.49036	0.50964	0.09744	0.71608	0.28392	0.10762
	STD	0.03871	0.03871	0.00494	0.04446	0.04446	0.00794
Metropolis Hasting(0.75)	Mean	0.49159	0.50841	18.84750	0.77099	0.22901	20.29062
	STD	0.01648	0.01648	0.42297	0.01580	0.01580	0.54866
Metropolis Hasting(0.85)	Mean	0.49459	0.50541	21.08563	0.75313	0.24687	22.83813
	STD	0.01648	0.01648	0.42057	0.01239	0.01239	0.56459
Metropolis Hasting(0.95)	Mean	0.48947	0.51053	23.64562	0.73586	0.26414	25.59750
	STD	0.02101	0.02101	0.40804	0.01361	0.01361	0.44475

Polytree with 15 nodes: $\alpha = 1$ and $\beta = 0.5$							
		Upsteam Evidence			Downstream Evidence		
Inference	Measure	PTrue	PFalse	Runtime	PTrue	PFalse	Runtime
Enumeration Ask	Mean	0.98080	0.01919	1.97614	0.93218	0.06781	2.16691
	STD	0	0	0	0	0	0
Likelihood Weighting	Mean	0.98116	0.01884	0.12713	0.91561	0.08439	0.12500
	STD	0.00428	0.00428	0.01005	0.04062	0.04062	0.00492
Gibbs Sampling	Mean	0.97104	0.02896	0.21869	0.94032	0.05968	0.22375
	STD	0.02086	0.02086	0.00494	0.04585	0.04585	0.00590
Metropolis Hasting(0.75)	Mean	0.97858	0.02142	42.69750	0.97095	0.02905	43.93187
	STD	0.00675	0.00675	0.93573	0.00574	0.00574	1.15825
Metropolis Hasting(0.85)	Mean	0.97942	0.02058	48.65563	0.96995	0.03005	49.45500
	STD	0.00478	0.00478	0.74704	0.00542	0.00542	0.82081
Metropolis Hasting(0.95)	Mean	0.98102	0.01898	54.38500	0.96943	0.03057	55.51937
	STD	0.00358	0.00358	0.91422	0.00621	0.00621	0.64569

Polytree with 25 nodes: $\alpha = 1$ and $\beta = 0.5$							
		Upsteam Evidence			Downstream Evidence		
Inference	Measure	PTrue	PFalse	Runtime	PTrue	PFalse	Runtime
Enumeration Ask	Mean	0.97200	0.02799	24.49636	0.80266	0.19733	87.49471
	STD	0	0	0	0	0	0
Likelihood Weighting	Mean	0.97108	0.02892	0.02892	0.80472	0.19528	0.19502
	STD	0.00436	0.00436	0.00767	0.01940	0.01940	0.00690
Gibbs Sampling	Mean	0.96344	0.03656	0.33977	0.80244	0.19756	0.34116
	STD	0.03874	0.03874	0.00679	0.08168	0.08168	0.00585
Metropolis Hasting(0.75)	Mean	0.96615	0.03385	66.42812	0.79433	0.20567	68.17625
	STD	0.00852	0.00852	1.24831	0.01550	0.01550	1.51761
Metropolis Hasting(0.85)	Mean	0.96623	0.03377	75.57750	0.79976	0.20024	76.75938
	STD	0.00587	0.00587	1.08378	0.01084	0.01084	1.16442
Metropolis Hasting(0.95)	Mean	0.96727	0.03273	84.55312	0.79852	0.20148	85.44688
	STD	0.00615	0.00615	0.95516	0.01613	0.01613	0.68072

4.4 Analysis

These results are quite intriguing. Every algorithm was able to predict PTrue and PFalse pairs within the +/-0.05 probability of exact inference.

4.4.1 Performance of Gibbs Sampling

On average, it seems that Gibbs sampling performed the worst, with the average predictions of PTrue and PFalse deviating quite a bit from the true PTrue and PFalse predictions from exact inference. There are cases from the tables where the mean predictions of PTrue and PFalse seem to match exact inference well. However, all of the predictions are accompanied with large standard deviations. These results are not surprising, as Gibbs sampling has a large range for sampling with all nonevidence variables and its associated markov blanket. This leads to a larger range in predictions which skew the results significantly.

4.4.2 Performance comparison between Likelihood Weighting and Metropolis Hasting

The comparison between Likelihood Weighting and Metropolis Hasting sampling is quite interesting.

For the given dataset, the average prediction of PTrue and PFalse made by Likelihood Weighting came the closest to the pairs made by exact inference. In addition, Likelihood weighting had the smallest runtime on average. This is a bit surprising, as Likelihood Weighting can suffer from decreased performance if it's making queries given downstream evidence variables with more evidence variables to consider. However, as seen in the polytree networks with 25 nodes, Likelihood Weighting predicted the best PTrue and PFalse pairs given

downstream evidence.

In contrast, Metropolis Hasting had the highest runtimes on average with the largest standard deviations as well with runtime. This is expected, as Metropolis Hasting had to run Gibbs sampling 100 times per sample. This also explains why as the probability of running Gibbs Sampling increased, the runtime of Metropolis Hasting increased. However, the prediction of PTrue and PFalse is more nuanced.

It seems for smaller polytree networks, the standard deviation in prediction between the two inference methods are comparable. However, for larger networks, Metropolis Hasting has smaller standard deviation. This is accompanied by cases where Metropolis predicts PTrue and PFalse relatively well compared to exact inference, such as the polytree networks with $\alpha = 1$, $\beta = 0.5$, and upstream evidence as well as polytree networks with uniform distribution and downstream evidence. However, Metropolis Hasting had poor predictions with polytree networks of $\alpha = 0.5$, and $\beta = 1$, especially for downstream evidence, so Metropolis Hasting seems to be useful only in select scenarios.

5 Conclusion

Bayesian networks are versatile and powerful structures for modelling random variables and their conditional independencies. Although there are exact inference methods for making predictions in the network, these methods are limited in scalability.

Based on the results from this paper, Likelihood Weighting is a safe choice for approximation inference for a polytree networks. However, for cases with specialized conditions such as uniform distribution with downstream evidence as well as CPT distribution trending towards one with upstream evidence, Metropolis Hasting could be a viable option. This is especially true if small standard deviations in the predictions on a large network is desirable. However, it should be noted that further experimentation with the probability parameter and Gibbs sampling per iteration needs to be considered to improve its performance and reduce the runtime in order to be viable alternative to Likelihood Weighting.

References

[1] Chad Crawford. gen bn. *URL: https://github.com/TUmasters/gen-bn*, 2018.

[2] Robert Geraghty. Exact inference. *URL: https://github.com/robger98/exact_inference*, 2021.

[3] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearsons, 4 edition, 2021.