
Machine Learning (CS 391L) : Spring 2013

Homework 4

Vineet Keshari (EID vk3226) mailto:vkeshari@cs.utexas.edu

---

1. Problem Description

The objective in the assignment is to perform Approximate Inference on a Bayesian Network using two sampling techniques: Rejection Sampling and Gibbs Sampling. In Approximate Inference, samples are generated from the joint probability distribution of the Bayesian Network. The samples can then be used to approximately answer any conditional query on the network. In this experiment, we generate a Bayesian Network of reasonable sophistication and evaluate several conditional queries on the network using both sampling techniques. We then compare the performance of the two techniques for several queries.

2. Network Structure

The Bayesian Network used in this experiment is shown in Figure 1. The network contains at least one example of the common structures found in Bayesian Networks that influence conditional independence (Downward Fork, V structure with child and nodes with zero, one and two parents). There is one diamond (X2,X4,X5,X8) and two X-structures (centered at X4 and X5). Node X5 has six nodes in its Markov Blanket.

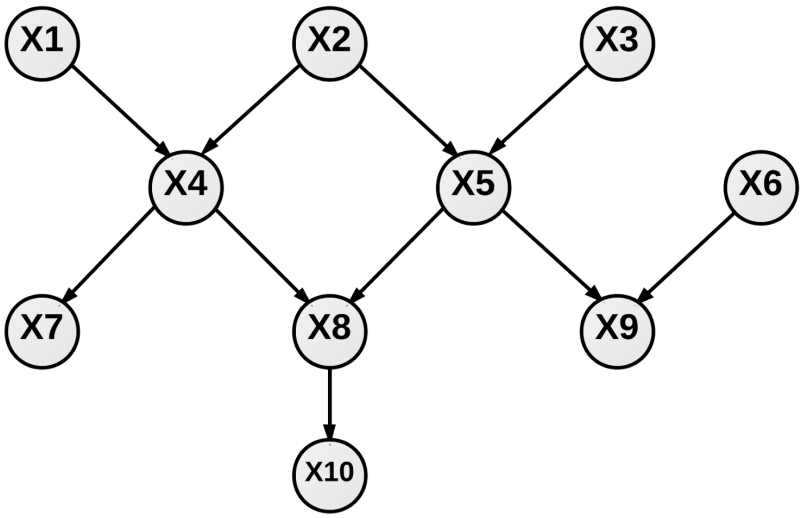

Figure 1: The Bayesian Network with 10 nodes used in this experiment.

The conditional probability for each node **`P[X|parents(X)]`** is generated randomly each time the algorithm is run. To load the last generated probabilities instead, set **`generate = 1`**. New queries with query and evidence nodes along with their respective values are randomly generated in each iteration for sampling. There is at least one query node in each query.

3. How to Run

On the Matlab command prompt, use the following commands:

```
>> generate = 1; % generate = 0 if you want to use the last saved network
>> run('hw4')
```

Note: The variable **`generate`** must be set before running the program.

You can set the following parameters in **`hw4.m`**:

`numQueries`: Total no. of queries to evaluate.

`numSamples`: Total no. of "useful" samples to generate for each sampling method.

`graph`: If set to 1, plots real-time data for sampling in addition to the error metrics.

4. Implementation

4.1. Rejection Sampling

Samples are generated using ancestral sampling. The nodes are evaluated in their topologically sorted order, so that the value of each variable can be generated conditioned on the already generated value of its parents. Samples that do not agree with the evidence are rejected. We continue evaluating until a target no. of samples have been accepted, or more than 100 times the target no. have been generated. The latter condition prevents sampling infinitely in case all samples are rejected.

4.2. Gibbs Sampling

Samples are generated by evaluating the probability for each variable conditioned on all other variables in the network, while keeping the evidence variables "clamped" to their given values. Once all variables have been generated, we evaluate the query using the sample. The first 10% of the target no. of iterations are ignored to allow for burn-in.

Figure 2 shows the performance of both Rejection and Gibbs sampling on a randomly generated query using 1000 samples. Of the 1739 samples generated during rejection sampling, 42.5% were rejected. In Gibbs sampling, there is no need to reject samples as all generated samples agree with the evidence. However, the first 100 out of 1100 samples were ignored to allow for burn-in. Each method therefore generated 1000 "useful" samples, providing a fair ground for comparison.

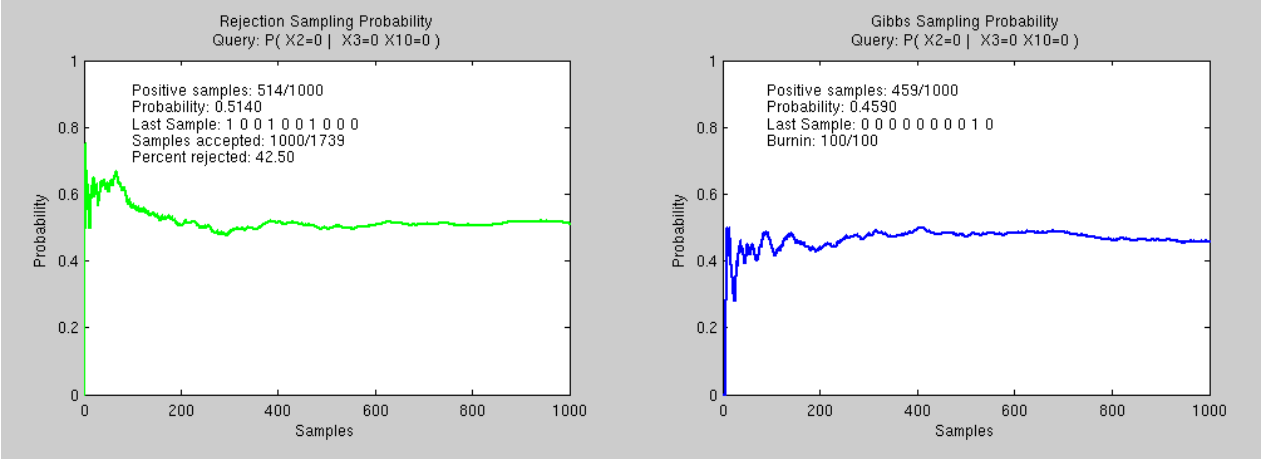

Figure 2: An example of approximate inference on the Bayesian Network shown in Figure 1 using Rejection (left) and Gibbs Sampling (right) for the query `P(X2=0|X3=0,X10=0)`.

6. Results

6.1 Comparison Error

We define the comparison error as the squared difference between the values evaluated using Rejection and Gibbs sampling. Since both of these techniques are approximate, they do not represent the true conditional probability of the query. Therefore, we do not evaluate the inference error w.r.t. the exact inference value in this experiment. Figure 3 shows the comparison errors over 100 iterations, with a new query randomly generated during each iteration, and 1000 "useful" samples generated for each query for both techniques. We see that the average comparison error between Rejection and Gibbs Sampling over 100 iterations is 0.0003. Since Rejection Sampling forms a baseline for our experiment and the comparison error is small, we can claim that our Gibbs Sampling works as expected.

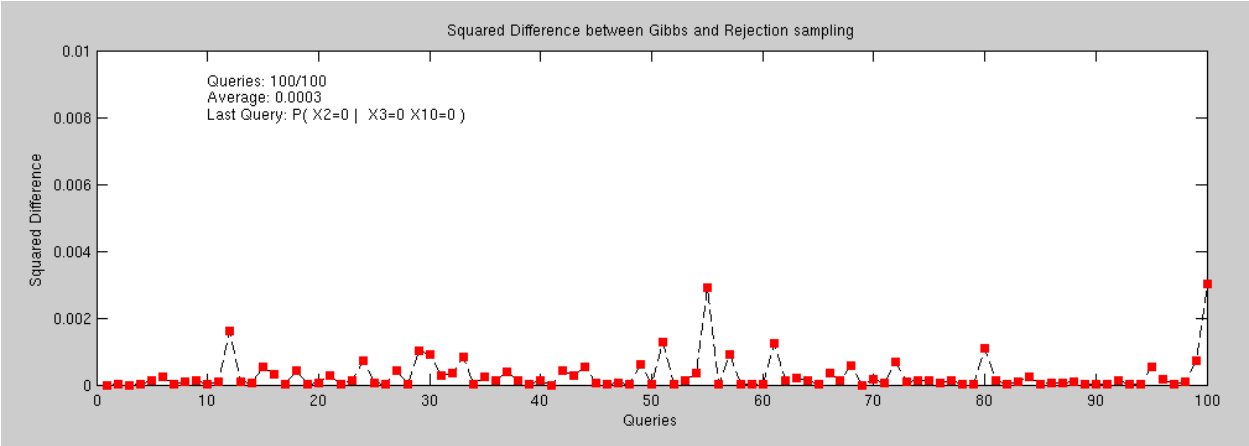

Figure 3: Comparison Error between Rejection and Gibbs Sampling for 100 queries on the same Bayesian Network. The graphs shown in Figure 2 are for the last of these queries.

6.2 No. of samples

Figure 4 shows the change in average comparison error as we increase the no. of samples for each query. We see that the comparison error drops sharply as the no. of samples increases from 100 to 700, and then decreases very slowly. We can see that the value is less than 0.001 for 400 or more samples, and less than 0.0003 for 900 or more samples, reinforcing our results from Section 6.1. For 1900 or more iterations, the comparison error converges towards a minimum value of less than 0.0001. This gives us an absolute error between the two techniques of less than 1%.

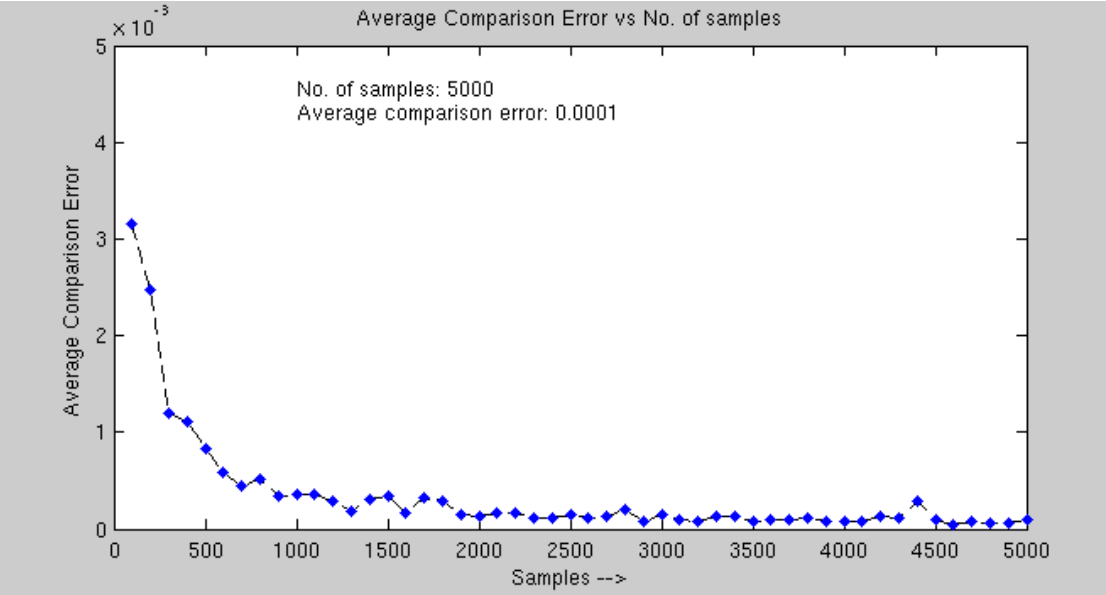

Figure 4: Average comparison error as no. of samples used during Approximate Inference increases. Results for each value of no. of samples have been averaged over 100 queries.

7. Discussion

While both sampling methods give similar results, Gibbs Sampling is much more efficient than Rejection Sampling, because it never generates samples that are not useful for evaluating the query. For some queries, over 99% of the samples were rejected during Rejection Sampling. On larger graphs, ancestral sampling can be very costly and a high rejection rate will significantly slow the algorithm down. Also, due to the unknown number of rejected samples, we cannot calculate an upper bound on the running time for a given no. of "useful" samples, which can theoretically be infinite. The performance of Gibbs sampling, on the other hand, can be bound as it is guaranteed to generate samples that agree with the evidence. Gibbs sampling can also be further improved by using a node's Markov Blanket for calculating its conditional probability, instead of all other nodes in the graph.