Mixture Model for Approximate Inference in Bayesian Networks

Peerapong Dhangwatnotai
Carnegie Mellon University
Pittsburgh, PA 15213
pdhangwa@andrew.cmu.edu

May 15, 2006

Abstract

Bayesian Networks (Bayes net) is a statistical tool for reasoning under uncertainty. However, exact inference in Bayes net is an NP-Hard problem. In this thesis, we propose an approximate inference algorithm which trains a mixture model to answer queries in a given Bayes Net. We use a large number of hidden classes for accuracy while taking advantage of the unlimited training data generated by the Bayes net. We compare the algorithm's query speed and accuracy with importance sampling on a real Bayes net.

1 Introduction

Bayesian network (Bayes net) is a statistical tool for modeling independences among a set of random variables. A Bayes net can compactly represent a joint distribution of a large number of variables. However, exact inference using Bayes nets is worst-case NP-hard [1]. Although approximate inference with error bound is also NP-hard [4], many approximation algorithms do very well in general. An example is a class of stochastic sampling algorithms which includes likelihood weighting, Gibbs sampling, importance sampling, etc. Stochastic sampling algorithms are also appealing because they converge to the correct answer as the number of samples goes to infinity.

However, stochastic sampling algorithms are still slow and require many samples in the case of unlikely evidences [3]. In this paper, we propose a mixture-model approximation algorithm which provides a fast inference and also converges to the correct answer in the limit. Then we compare the algorithm's accuracy and inference speed with importance sampling on a real Bayes net.

2 Bayesian Networks

Bayes net is a language for representing the joint probability distribution of a set of random variables $\{X_1, X_2, \ldots, X_n\}$. A Bayes net consists of a directed acyclic graph and a set of conditional probability distributions which are usually stored as conditional probability tables (cpts). Each node in a Bayes net corresponds to a random variable and thus we call each node by the name of its corresponding random variable. Let X_i be a random variable in a Bayes net. The set of parents of X_i, denoted $\{\pi_i\}$, is the set of random variables whose nodes are parents of X_i. The graph encodes independence assumptions of the form: X_i is independent of its non-descendants given its parents, $\{\pi_i\}$. There is a cpt associated with each node X_i to specify the probability of X_i given its parents, $\mathrm{p}(X_i|\{\pi_i\})$. The joint probability distribution is specified by $\mathrm{p}(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} \mathrm{p}(X_i|\{\pi_i\})$.

3 Mixture Model

A mixture model is a Bayes net with one special random variable called "class." The only independence assumption that a mixture model encodes is all variables are independent given class. A picture of a mixture model is shown in figure 1. The structure of the network is the same as naive Bayes but the reason we call it mixture

model is because during training, class is an unobserved variable.

The idea behind the algorithm is we can model a Bayes net with a mixture model more and more accurately as the number of classes increases. With enough number of classes, the mixture model can precisely model a Bayes net. This allows us to make a trade off between training time and accuracy. The next two subsections deal with mixture model training and inference.

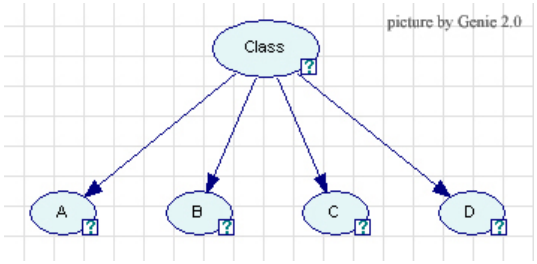

Figure 1: Mixture model

3.1 Training

Before doing an inference, first we must train a mixture model to approximate a given Bayes net. Since training is a one-time cost, we can afford to spend huge amount of time on training. Two things that affect the accuracy of inference the most are number of classes and number of samples. More classes and more samples lead to better accuracy. During training, we fix a number of classes and a set of samples. The samples are generated from the Bayes net we want to model.

The standard approach for training a Bayes net with unobserved variables is to use EM algorithm. On the other hand, training a mixture model is equivalent to clustering the samples and in here, we will explain the training algorithm in terms of clustering. Algorithm 1 shows the pseudo code. Doing EM in a discrete space is different from a continuous space in that the samples appear as point mass and there is no relationship between any two values of an attribute. For example, it is possible to choose a sample as an initial location for a class such that it lands on top of an existing class. Therefore, we have to modified the standard EM algorithm for this purpose.

When initializing class locations, we do a 2-pass assignment. In the first pass, we choose a sample at random as an initial location for each class. In the second pass, one by one, we choose a sample such that for each preceding class, the probability that it is in the class is less than a certain threshold. It is possible that there is no such sample. Therefore, we have to be less picky over time. This is accomplished by increasing the threshold each time we draw a disqualified sample. The initial location of a class is the average between the samples assigned in the first pass and the second pass.

During the E-step in the EM, we assign a sample to at most two classes. First we find the two most likely classes. If the probability that the sample is in the most likely class is greater than the probability that it is in the second most likely class by more than a constant (0.2), we assign the whole sample to the most likely class. Otherwise, we assign a fraction of the sample to each of the two classes according to the ratio of their probabilities. This speeds up the EM when we have many classes and prevents more than two classes from staying at the sam e location.

It is possible that during the course of an EM, a class is "starved". We define "starved" as having number of samples less than a certain threshold. When a class is starved, we relocate the center of the class to a random sample with some criteria and set the probability or weight of the class to (number of samples)/(number of classes). Since it is possible to choose a sample which is on top of an existing class, some criteria are needed to make sure the class is not likely to become starved again. To do that, we calculate the probability that the sample is in each class and require that none of the probabilities exceeds a certain threshold.

We have observed that it is possible to use only unique samples and assign each sample a different weight proportional to its probability. Using unique samples will help speed up training and greatly decrease the probability of choosing two samples of the same location. However, it is an item of future research because there is not enough time to reimplement the EM.

3.2 Inference

Inference is the process of calculating the probability of some variables given some variables. The strength of mix-

Algorithm 1 pseudo code for EM

```
Initialize class locations
repeat
  E-step: Assign samples to most likely class(es)
  M-step: Estimate parameters p(variable|class) from the assigned samples
  for each starved class do
    repeat
      Pick a sample, μ
      Increase δ by a fraction
    until for each class except the starved class, p(sample|class) is less than δ
    Initialize the class with mean μ and some amount of variance.
    Set the weight (prior) of the class to 1/(number of classes).
  end for
until parameters of the mixture model change by less than a certain threshold
```

ture model is running time for inference is linear in the number of classes. This is shown in the inference formula below.

$$p(X|Y) = \sum_C p(X, C|Y) \quad (1)$$

$$= \sum_C p(X|C, Y)p(C|Y) \quad (2)$$

$$= \sum_C p(X|C)p(C|Y) \quad (3)$$

Note that the assumption that X and Y are independent given C (class) is the mixture model assumption which will be true if the number of classes equals the number of possible values of the joint distribution of all variables. In that case, we can let each class contain only samples of the same configuration and thus all variables are independent.

4 Experiment

4.1 Convergence test

First, we confirmed that in the limit, the mixture model converges to the true Bayes net distribution. We trained a mixture model on a small Bayes net called fash net as shown in the figure 2. We varied the number of classes from 2 to 16 and used two sets of samples, one of size 10000 and the other of size 100000. Then the difference between the mixture model distribution and the Bayes net distribution is measured using KL-divergence. Specifically, let P be the Bayes net distribution and Q be the distribution modeled by the mixture model. We want to estimate

$$D_{KL}(P||Q) = \sum_x P(x) log(\frac{P(x)}{Q(x)})$$

This is accomplished by drawing 10000 samples from the fash net and calculate $\frac{1}{n}\sum_{i=1}^{n} log(P(x_i)/Q(x_i))$ where $P(x_i)$ is the probability of sample i given by the Bayes net and $Q(x_i)$ is the probability of sample i given by the mixture model. It can be shown that this is an unbiased estimate of $D_{KL}(P||Q)$. Figure 3 shows the result which suggests that as the number of classes and the number of samples increase, the mixture model's distribution converges to the Bayes net distribution. Modeling the fash net with 16 classes and 100000 samples, we have KL-divergence as low as 0.000061.

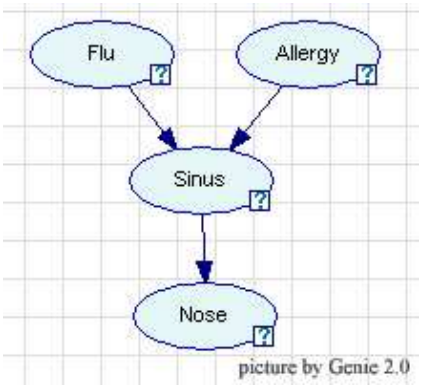

Figure 2: fash net

4.2 Query Test

To see how well mixture model does in real world networks, we compared its accuracy and speed with importance sampling on a real network, called alarm network.

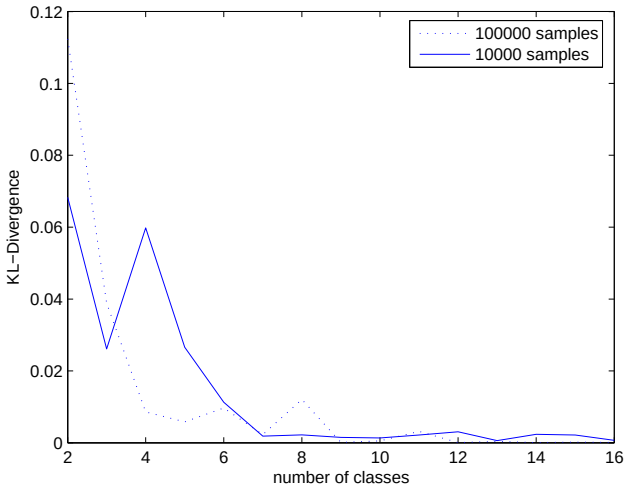

Figure 3: Mixture model convergence

The network, shown in figure 8, is used by medical experts to monitor patients in intensive care units.

Mixture models with number of classes from 10 up to 310 were trained with 100000 samples. The KL-Divergence between the mixture model and the Bayes net decreases as the number of classes increases (shown in figure 4). Since the mixture model with 310 classes has the best KL-divergence, it is used in the performance comparison with importance sampling.

To test query performance, we generated 2000 queries by drawing 2000 samples from the alarm network and for each sample, randomly set 10 of the 37 variables to be missing. Therefore, a query consists of 10 subqueries, which are, for each missing variable, p(the missing variable|evidence). The evidence is simply the variables whose values are not missing. Since there are only 10 missing variables, we can calculate the correct answers using exact inference.

Accuracy

To compare the accuracy, we calculated the score for each algorithm based on KL-divergence. For each subquery, we calculated the KL-divergence between the distribution given by an approximate inference (mixture model or importance sampling) and the correct distribution. The

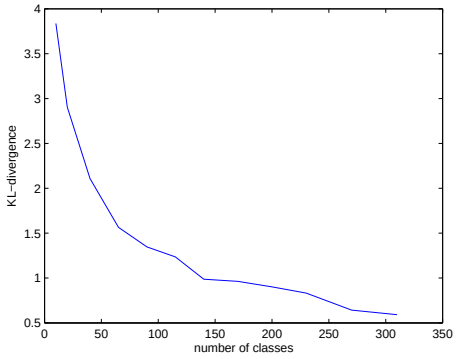

Figure 4: Alarm network convergence

score an algorithm gets for each query is the average KL-divergence of the subqueries. Each algorithm's final accuracy score is the average score over all queries. Since KL-divergence measures the difference between two distributions, the lower the score, the better the accuracy.

Importance sampling is very accurate in general. However, there were some queries in which the probabilities of some variables taking on some particular values were

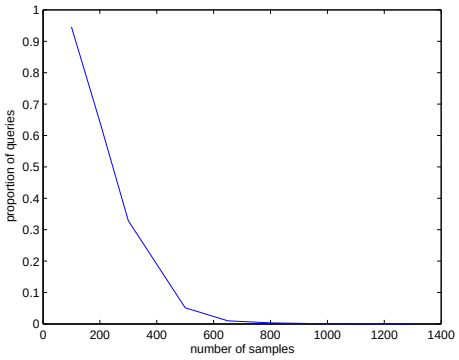

Figure 5: Proportion of queries which the number of samples is insufficient for importance sampling to determine the distribution

0 because importance sampling did not see any sample of those values. Therefore, the KL-divergence of importance sampling for those queries are infinite or undefined. Figure 5 shows the proportion of queries which cannot be answered with a certain number of samples. Importance sampling requires at least 1000 samples to be able to answer all queries.

Excluding the queries which importance sampling could not answer, we obtained the score for importance sampling as shown in figure 6. Mixture model's score is 0.03382.

Speed

Inference speed was measured by timing each algorithm on the same set of queries and computing the average time per query. The result is shown in figure 7.

5 Discussion

When running time is not a concern, importance sampling can draw more samples and achieves accuracy beyond mixture model. However, when we want fast inference, drawing a few hundred samples is not an option for importance sampling because it will not have a sample for some values of some attributes. As shown in figure 5, importance sampling needs at least 500 samples to be able to answer at least 95% of the queries. Drawing the amount of samples, importance sampling takes about 5 times longer than mixture model to answer a query.

Even if we ignore the insufficient sample problem, mixture model still has the advantage when time is a factor in performance evaluation. Importance sampling requires between 150 and 200 samples to achieve the same level of accuracy as the mixture model. When using that amount of samples, importance sampling is 1.45 to 2 times slower than mixture model. If we fix the running time at mixture model's speed, mixture model's accuracy score is 27.43% more than importance sampling's.

The above statistics are specific to the network and the type of query. There is more research to be done to examine the generalizability of this result. It is interesting to see how both algorithms perform on other networks and with a different number of missing variables. We conjecture that if there are more missing variables, importance sampling will take longer time to converge because the space to sample from is larger while mixture model will still take about the same amount of time. It is also possible to combine the two algorithms together. We can use the distribution from a mixture model as a proposal distribution for importance sampling. This will help importance sampling converge faster.

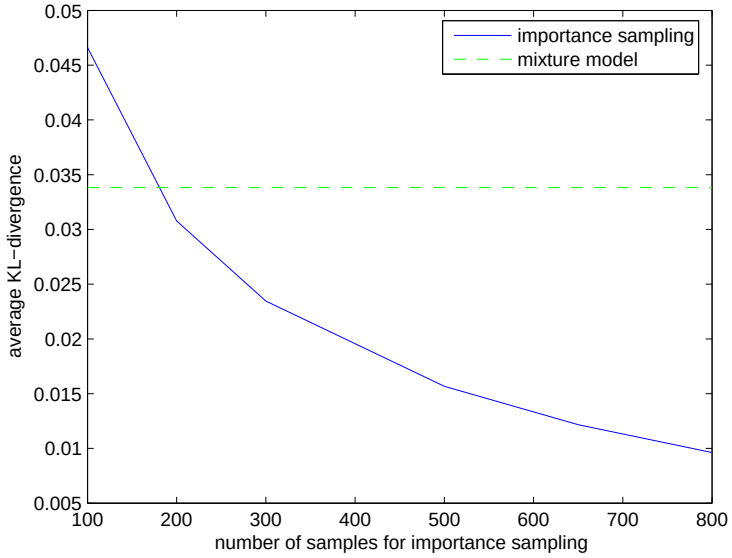

Figure 6: Accuracy score

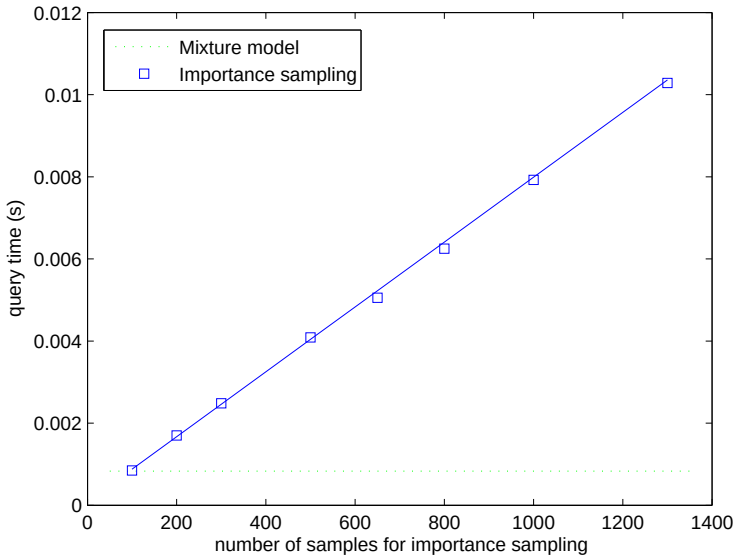

Figure 7: Inference speed

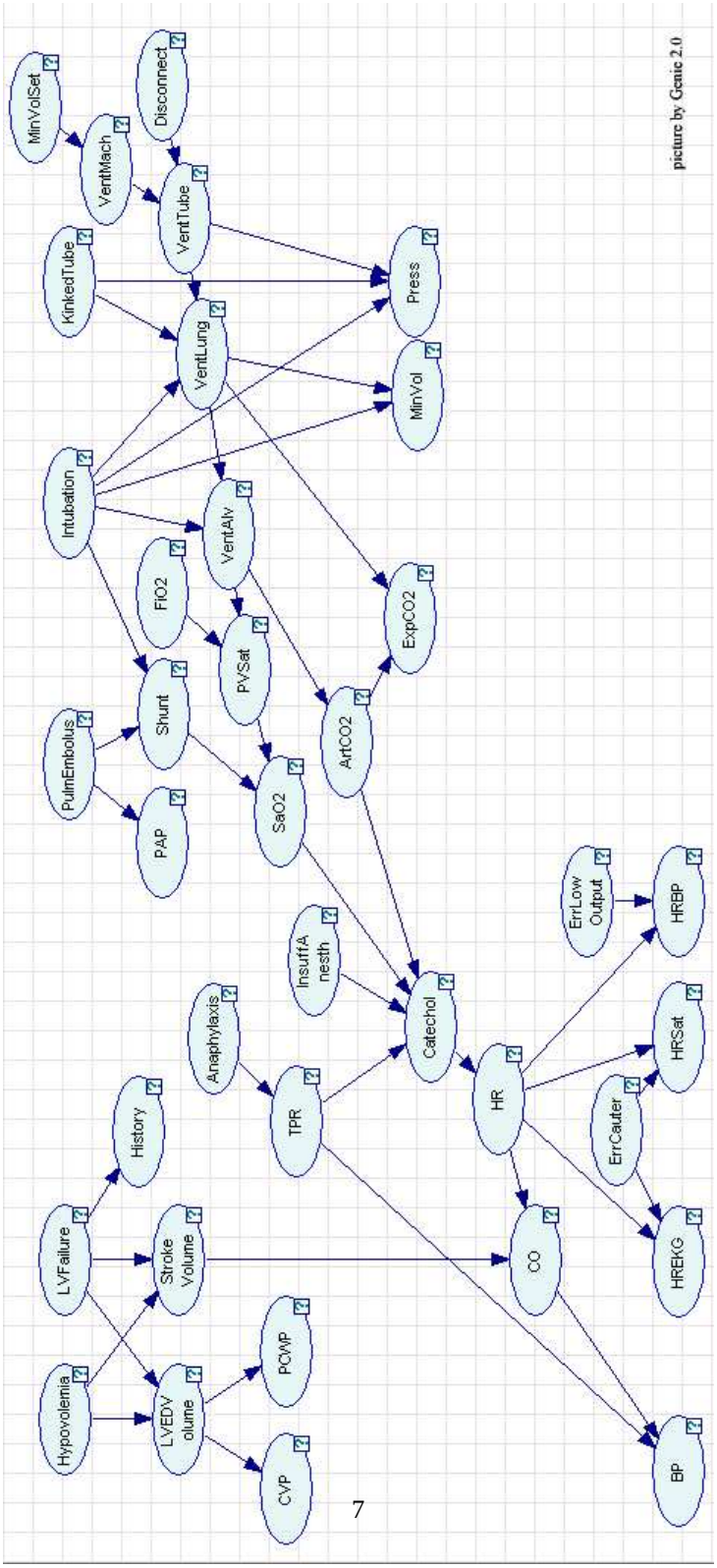

Figure 8: Alarm network

References

[1] G. F. Cooper (1990) The computational complexity of probabilistic inference using Bayesian belief networks, *Artificial Intelligence*, 42(2-3):393-405.

[2] D. Lowd, and P. Domingo (2005) Naive Bayes models for probability estimation, *Proceedings of the 22 nd International Conference on Machine Learning*, Bonn, Germany.

[3] C. Yuan and M. J. Druzdzel (2003) An Importance Sampling Algorithm Based on Evidence Pre-propagation, *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence*, Pittsburgh, Pennsylvania.

[4] P. Dagum and M. Luby (1993) Approximating probabilistic inference in Bayesian belief networks is NP-hard, *Artificial Intelligence*, 60(1):141-153.

[5] Genie developers (2006) http://genie.sis.pitt.edu/.