Implementation and Optimization of Variable Elimination Algorithm of Bayesian networks

Zijian Liu(#37689570), Pengyu Ji(#21364874), Xiang Yan(#68687977)
Department of Computer Science
University of California, Irvine, CA, 92697

Abstract

A Bayesian network is a probabilistic graphical model that represents dependencies between a set of random variables. In this report, we implement the variable elimination algorithm, one of the most popular interference-based Bayesian network processing algorithms, and apply the program on given problem data sets. Result shows only a 50% completeness after 24hrs of running on our simplest data set. After this, we optimize the original variable elimination algorithm by ordering the non-evidence data using min-fill heuristic before applying variable elimination. Result shows a significant improvement in running time after algorithm's optimization.

1. Introduction

1.1 Bayesian network

A Bayesian network is a graphical model that represents dependencies between a set of random variables. It is also known as recursive graphical models, bayesian belief networks, belief networks, causal probabilistic networks, causal networks, influence diagrams and many other more. Bayesian networks not only provide guides to systematically structure a situation’s probabilistic information into a directed acyclic graph (DAG), but also provide a set of algorithms that allow people to derive implications from those constructed information, which can eventually be used in the conclusions or decisions making of the corresponding situation[1]. Bayesian networks have been drawn significant attention over the last few decades across a large number of fields in both academia and industry. Successful

applications of Bayesian networks play huge role of diagnosis, troubleshooting, data mining, pattern recognition, decision making, and many other problems in a variety of domains[1].

Every Bayesian network can be constructed as a directed acyclic graph G(**X**, **E**) where **X** = {X1, . . . , Xn} is a set of random variables and **E** is a set of directed edges. Each node in a Bayesian network is associated with a conditional probability table (CPT), whose functional form is P(Xi|pa(Xi)).

1.2 Processing algorithms for Bayesian network

Algorithms for processing graphical models fall into two general categories: inference-based algorithms and search-based algorithms.

Inference-based algorithms (e.g., Variable Elimination, Tree Clustering) are better at exploiting the independencies captured by the underlying graphical model[2].

While, search-based algorithms (e.g., depth-first branch-and-bound, best-first search) traverse the model’s search space where each path represents a partial or full solution. The linear structure of search spaces does not retain the independencies represented in the underlying graphical models and, therefore, search-based algorithms may not be as effective as inference-based algorithms in processing this property[2].

1.3 Define the problem

Our goal is simply to implement the Bayesian networks' variable elimination algorithm and apply the program on given problem data sets. We use P(e) problem instances provided by the course website[3] as our test data sets. On those data sets, a number of variables and their conditional probability tables (CPTs) which indicate the relations among each variable are provided. Each variable has its own domain size. We try to find the probability that the evidence happens by applying variable elimination algorithm. Given the

evidence of variables, our algorithm is to calculate the probability of this event based on the dependency, value domain and CPTs.

Variables each has its own domain size. Each variable v(i, d) in V defines its properties. i denotes the ID of variable and d denotes the domain size of it. Domain range is [0, 1, 2, ... ,d].

Evidence e(i) in E defines the value of evidence variable. e(i) can only be value within domain of v(i, d).

Dependency d(i) denotes the dependency of variable v(i,d), it shows the variables that probabilities of v(i,d) changing based on.

Conditional Probability Tables CPT(i) shows the probability table of v(i, d), every probability is ordered according to the variable in d(i).

2. Processing Approach

2.1 Evidence Condensing

The first step to start the whole processing procedure is to condense the conditional probability tables. The main algorithm is, for each CPT which contains one or more evidence variables, instantiate the evidence and remove that evidence variables from the original CPT to form a new condensed CPT. An example of condensing CPT is given in figure 1. Evidence B is known to be 1 in the shown example and left chart (original CPT) is associated with B, thus the original CPT can be condensed into a new smaller CPT shown as the right chart.

A	B	Probability
0	0	0.3
0	1	0.2
1	0	0.4
1	1	0.1

Evidence B = 1 →

A	Probability
0	0.2
1	0.1

Figure 1. Example of condensing conditional probability table

2.2 Original Variable Elimination Approach

Variable Elimination[4] is the key function that contains the key algorithm. This algorithm eliminates each variables by multiplying them.

Let Φ be the set of CPTs. o = {X1, ..., Xn} be the set of non-evidence variables. The main algorithm is follow:

For i = 1 to n:

Multiply all CPTs in Φ which contains Xi;

Sum-out Xi from it yielding a function φ_{new}*;*

Remove all functions that contain Xi from Φ;

add φ_{new} *to Φ.*

The multiply operation is shown in figure 2. It is an algorithm to combine two CPTs.

The sum-out of a variable Xi from a function φ(X) such that Xi ∈ X is a function φ′ defined over the set of variables X′ = X \ {Xi}. The value of each assignment x′ ∈ D(X′) is given by:

$$\varphi'(x') = \varphi(c(xi, x'))$$

where c(xi, x′) denotes the composition of the assignments xi and x′. Figure 3 shows the sum-out operation.

A	B	Φ(a, b)
0	0	0.3
0	1	0.2
1	0	0.2
1	1	0.3

*

B	C	Φ(b, c)
0	0	0.1
0	1	0.4
1	0	0.3
1	1	0.2

=

A	B	C	Φ(a, b, c)
0	0	0	0.06
0	0	1	0.24
0	1	0	0.12
0	1	1	0.08
1	0	0	0.04
1	0	1	0.16
1	1	0	0.18
1	1	1	0.12

Figure 2. multiply operation

$\sum a$

A	B	Φ(a, b)
0	0	5
0	1	7
1	0	3
1	1	4

=

A	Φ′(b)
0	5+3=8
1	7+4=11

Figure 3. sum-opt operation

2.3 Optimized Approach

We realize that performing the variable elimination right after evidence condensing is very costly. Since only a 50% completeness was achieved after 24hrs of running on our simplest data set. The reason is simply because multiplication of two large arrays is very time consuming. We try to solve this problem by ordering the non-evidence variables using min-fill heuristic before performing variable elimination.

The main idea of min-fill heuristic is to order variables with their relation complexity by ascending order. Variables with the least connections with other variables will be eliminated first. We judge their relation according to the edges need to be added when eliminated. The algorithm is follows:

For i = 1 to n:

Select a variable X which when eliminated will add the least number of edges in the induced graph;

Remove X from the graph ;

Add X to position i;

Link the neighbors of X which are not yet linked;

Return the ordered queue.

The way to eliminate a variable from a graph is shown in figure 4. In figure 4, after eliminating A from the graph, all A's neighbors which are not yet linked are linked, e.g. (B, C) and (B, D).

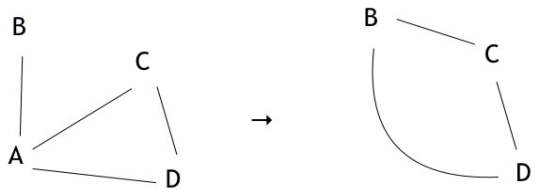

Figure 4. Eliminate A from a graph, link the neighbors of A which are not yet linked

3. Problem and Discussion

First, we used original variable elimination approach to construct our program, and tried it out on the given data sets. Unfortunately, the process of non-evidence variable elimination without ordering is extremely time costing. Only a 50% completeness was achieved after 24hrs of running on our simplest data set.

Then, we were forced to apply the optimized algorithm on our program. At this time, we were able to finish running the tests. The running time was improved dramatically in our optimized program. It shows the importance of a good heuristic.

Comparison shows in Table 1. Both program (original program and optimized program) was running on a machine with memory 8 GB 1600 MHz and processor 2.6 GHz Intel Core i5. Since the original running time is long, we just test this algorithm on some small scale test data.

Table 1. Comparison of original variable elimination approach and optimized method

Data File Name	Number of Variables	Number of Evidence	Number of Arcs	Running Time (Original)	Running Time (After optimization)	Result
BN_0.erg	100	26	300	Over 24 hours	5244 msc	1.28E-9
BN_1.erg	100	18	394	-	668996 msc	6.65E-7
BN_3.erg	100	36	451	Over 24 hours	2830 msc	2.76E-13
BN_4.erg	100	51	574	Over 24 hours	56 msc	3.59E-18
BN_5.erg	125	55	678	Over 24 hours	4978 msc	1.84E-19
BN_6.erg	125	71	948	Over 24 hours	112 msc	4.29E-26
BN_7.erg	95	30	535	-	493191 msc	9.63E-8
BN_10.erg	85	17	304	-	11503 msc	6.24E-6
BN_11.erg	105	46	631	-	178487 msc	7.96E-18

As Table 1 shows, the original running time is always over 24hrs while the optimized algorithm's running time is much better than original time. It can be concluded that the min-fill heuristic ordering plays significant roles in the improvement.

4. Conclusion

The Variable Elimination Algorithm of Bayesian Networks with min-fill heuristic performs well in medium size data. It can get the result in a feasible time. .....

In this paper we introduce and implement the Variable Elimination Algorithm of Bayesian Networks. The order of non-evidence variables causes huge difference on time and space cost. After that, we introduce the min-fill heuristic function for ordering of non-evidence variables. Based on the experiment result, it tremendously reduces the time cost of performance. However, when the number of variables and dependencies increase, the cost for calculation becomes exponential. In the future, we will implement AND/OR search tree and other algorithms for Bayesian Network in order to calculate the exact probability of Bayesian Network.

Reference

[1] Adnan Darwiche; Communications of the ACM, Vol. 53 No. 12, Pages 80-90, 2010.

[2] Rina Dechter, Robert Mateescu; Artificial Intelligence; Vol. 171, Issue: 2-3, Pages: 73-106, 2007

[3] Problem repository: http://graphmod.ics.uci.edu/repos/pe/uaicomp06/

[4] Vibhav Gogate; Bayesian Networks: Representation, Variable Elimination; http://www.hlt.utdallas.edu/~vgogate/ml/2012s/notes/BN_notes.pdf