Journal of Machine Learning Research 1 (2021) Submitted ; Published NA

Bayesian Inference

Cooper Strahan COOPERSTRAHAN@GMAIL.COM
Gianforte School of Computing
Montana State University
Bozeman, MT 59715, USA

Riley Slater RILEYMAIL6@GMAIL.COM
Gianforte School of Computing
Montana State University
Bozeman, MT 59715, USA

Editor:

Abstract

In order to make fast decisions about an environment in AI, decisions need to be made based on probabilities. Given a Bayesian Network, a variable, and evidence, marginal distributions need to be found in order to determine a most likely scenario. In this project we generated two algorithms exact and approximate to determine marginal distributions for a given variable based on a Bayesian Network and some evidence. By determining these distributions our agent will be able to make decisions on a high probability scenario, this will give agents the ability to make faster decisions based on their surroundings.

Keywords: Bayes Rule, Bayes Net, Bayesian Inference, Exact Inference, Gibbs Sampling

1. Introduction

In this project we parsed BIF files into a Bayesian Network. The team then ran Exact Inference and Gibbs Sampling algorithms on the network to determine the marginal distribution with different levels of given evidence. The marginal distributions were then written to an output file.

2. Problem Statement and Hypothesis

The problem addressed in this paper is to determine the best method in order to approximate a marginal distribution of a variable given some evidence. Two algorithmic approaches were taken. Exact Inference, which if implemented correctly should always return the correct solution, and Gibbs Sampling which returns an approximate solution for problems that are slow to run. (Cooper, 1990) proved Exact Inference to be an NP-Hard problem meaning that its implementation can be exponential to run [Coo90].

2.1 Hypothesis

The Exact Inference algorithm will be very slow to run, as enumerating all of the combinations of values in the Bayesian Network is an exponential process. The Exact Inference

Algorithm will return the correct solution. Gibbs Sampling will be much faster, as it is randomly iterating around the board, and will return close to the correct results.

3. Algorithm Description and Implementation

Our implementation consisted of two parts, parsing the data into a Bayesian Network, and running both the Exact Inference and Gibbs Sampling Algorithms on that Bayesian Network.

Our Bayesian Network was a list of Bayesian nodes. The Bayesian Node objects, maintained the node name, type (discrete or continuous), number of Types, types (True, False; Low, Med, High; etc.), the marginal distribution, a list of node parents, list of node children, and the conditional probability tables (CPT).

Our parsing algorithm read in BIF files, creating Nodes, when a creation statement was present and appending CPTs and other information to the already created nodes once the CPTs were reached. After network generation the parser then used (Kahn's, 1962) topological sort algorithm to sort the network [Kah62].

The Exact Inference algorithm followed the pseudo-code from the Elimination Ask algorithm in the textbook. The Elimination Ask algorithm accepts three parameters, X - the variable whose marginal distribution we are searching for, e - the evidence provided to the algorithm, bay_net - the bayesian network that is being used to determine marginal distributions. An empty dictionary of factors was created, the factor represents a simplified node containing multiple variables and a CPT. Our algorithm, in order to improve time complexity took evidence variables, iterated through the network and restricted the CPTs of evidence variables to only include the given evidence. Including this pre-processing step reduces the complexity of the joined factor tables later on in the algorithm. It also reduces the node types, effectively including the restrictions when parents are examined in the sum out and point-wise product functions described later. The algorithm then reversed the topologically sorted nodes, in order to improve algorithm speed, per the textbook. The nodes are then iterated through. During the iteration process, the node is first turned into a factor, where it is transitioned from a node in form

P(A | B, C) = (True, True) 0.2 0.4 0.4,
(True, True) 0.2 0.6 0.2,
....
Into an object in this form.
{(A,B,C): {(True, False): [0.2, 0.4, 0.4], ...}

The current variable A is checked to determine whether or not it is a hidden variable. A hidden variable is a variable that is not the X variable in question or a variable included in the given evidence. If a variable is hidden, the variable is summed out. The sum out process works by accepting a variable V, a factor array factors, and a bayesian network bay_net. It creates two new factors F' and F", and returns a new factors array including F". F' contains all of the variables in the current list of factors that contain the variable to be summed out. F' is generated by enumerating all of the potential combinations of value types that exist for each of variables in the new factor. If we were combining A, B and C, and all their types were True and False. We would generate the values (True, True, True), (True, True, False),

..., (False, False, False). This is the process that slows down the algorithm as enumerating very large combinations of variables is exponential dependent of the number of variables and their types. The point-wise product of these values is then taken to generate F'. The process to generate a point-wise product in our algorithm, searches for matching type values from the original variables appending them to a corresponding array. Meaning if we are looking to find the values for (True, True, True), we would take the value corresponding to A = True, B= True and the value corresponding to B = True and C = True, and multiply these two values together. This process is then repeated to fill the F' dictionary for all enumerated type values. Once F' has been generated another factor F" is created. F" contains all of the variables except for the variable that is being summed out. The variable types are then enumerated to fill a dictionary for F". We then perform a similar operation to the point-wise product, pulling corresponding values from F' to fill the F" table. Rather than add the values together the values are summed. A single instance of F" then replaces all of the values from the factors array that contained the variable to be summed out. The new factors array then replaces the old factors array. This process is repeated for every variable in the network. Once this process is complete, we are left with only the X variable. The X variable then has the point-wise product taken, and is normalized such that all of the values in its marginal distribution sum to 1. We then return the value of its marginal distribution.

Our implementation of Gibbs Sampling was based on the textbook's Gibbs Ask algorithm [RN22]. The algorithm works by accepting the evidence, a bayesian network bay_net, and a number of iterations num_iter. The algorithm returns a marginal distribution based on a given variable. The algorithm first iterates through the list of evidence as well as the bay_net, setting the type of the values in the network to the given evidence type. This allows the algorithm to reduce the space search when examining variable values by ignoring the types in the variable not equivalent to the current type. The algorithm then iterates num_iter times, we made a design decision to set this number to 10,000. The algorithm selects a random variable Z_i inside the network and calculates Z_i's markov blanket. A variable's markov blanket consists of all of its children, parents, and children's parents (excluding itself). Variable Z_i and the rest of the markov blanket has all of their types asserted onto them, and the probability of these asserted types returned to the inference engine. Types are only asserted on non-evidence variables. We then assign a marginal distribution for Z_i based on the returned values from the random selection. After performing all of the iterations, we then return the marginal distribution of the variable X.

4. Experimental Approach

Our experimental approach involved running the exact inference once each for each combination of search variable and level of evidence. We ran the Gibbs Sampling algorithm 5 times for each combination. After analyzing the data, we wrote the most consistent results returned from the algorithm rather than the averages. As we felt that it better represented the solutions found by the algorithm.

5. Results

Table 1: Table to describe the results from the exact inference study. The exact inference algorithm was too slow to complete and so the majority of the results are missing

Dataset	Variable	Evidence	Marginal Distribution
alarm	HYPOVOLEMIA	N/A	[]
alarm	LVFAILURE	N/A	[]
alarm	ERRLOWOUTPUT	N/A	[]
alarm	HYPOVOLEMIA	Little	[]
alarm	LVFAILURE	Little	[]
alarm	ERRLOWOUTPUT	Little	[]
alarm	HYPOVOLEMIA	Moderate	[0.5, 0.5]
alarm	LVFAILURE	Moderate	[]
alarm	ERRLOWOUTPUT	Moderate	[]
child	Disease	N/A	[0.11, 0.32, 0.27, 0.19, 0.05, 0.05]
child	Disease	Little	[0.11, 0.32, 0.27, 0.19, 0.05, 0.05]
child	Disease	Moderate	[0.11, 0.32, 0.27, 0.19, 0.05, 0.05]
hailfinder	SatContMoist	N/A	[]
hailfinder	LLIW	N/A	[]
hailfinder	SatContMoist	Little	[]
hailfinder	LLIW	Little	[]
hailfinder	SatContMoist	Moderate	[]
hailfinder	LLIW	Moderate	[]
insurance	MedCost	N/A	[]
insurance	ILiCost	N/A	[]
insurance	PropCost	N/A	[]
insurance	MedCost	Little	[]
insurance	ILiCost	Little	[]
insurance	PropCost	Little	[]
insurance	MedCost	Moderate	[0.96, 0.02, 0.01, 0.007]
insurance	ILiCost	Moderate	[0.95, 0.02, 0.02, 0.01]
insurance	PropCost	Moderate	[]
Win95pts	Problem1	N/A	[]
Win95pts	Problem2	N/A	[]
Win95pts	Problem3	N/A	[]
Win95pts	Problem4	N/A	[]
Win95pts	Problem5	N/A	[]
Win95pts	Problem6	N/A	[]
Win95pts	Problem1	Problem1	[]
Win95pts	Problem2	Problem1	[]
Win95pts	Problem3	Problem1	[]
Win95pts	Problem4	Problem1	[]
Win95pts	Problem5	Problem1	[]
Win95pts	Problem6	Problem1	[]
Win95pts	Problem1	Problem2	[]

Win95pts	Problem2	Problem2	[]
Win95pts	Problem3	Problem2	[]
Win95pts	Problem4	Problem2	[]
Win95pts	Problem5	Problem2	[]
Win95pts	Problem6	Problem2	[]
Win95pts	Problem1	Problem3	[]
Win95pts	Problem2	Problem3	[]
Win95pts	Problem3	Problem3	[]
Win95pts	Problem4	Problem3	[]
Win95pts	Problem5	Problem3	[]
Win95pts	Problem6	Problem3	[]
Win95pts	Problem1	Problem4	[]
Win95pts	Problem2	Problem4	[]
Win95pts	Problem3	Problem4	[]
Win95pts	Problem4	Problem4	[]
Win95pts	Problem5	Problem4	[]
Win95pts	Problem6	Problem4	[]
Win95pts	Problem1	Problem5	[]
Win95pts	Problem2	Problem5	[]
Win95pts	Problem3	Problem5	[]
Win95pts	Problem4	Problem5	[]
Win95pts	Problem5	Problem5	[]
Win95pts	Problem6	Problem5	[]
Win95pts	Problem1	Problem6	[]
Win95pts	Problem2	Problem6	[]
Win95pts	Problem3	Problem6	[]
Win95pts	Problem4	Problem6	[]
Win95pts	Problem5	Problem6	[]
Win95pts	Problem6	Problem6	[]

Table 2: Table to describe the results from the approximate inference study.

Dataset	Variable	Evidence	Marginal Distribution
alarm	HYPOVOLEMIA	N/A	[0.2, 0.8]
alarm	LVFAILURE	N/A	[0.05, 0.95]
alarm	ERRLOWOUTPUT	N/A	[0.05, 0.95]
alarm	HYPOVOLEMIA	Little	[0.2, 0.8]
alarm	LVFAILURE	Little	[0.05, 0.95]
alarm	ERRLOWOUTPUT	Little	[0.05, 0.95]
alarm	HYPOVOLEMIA	Moderate	[0.2, 0.8]
alarm	LVFAILURE	Moderate	[0.05, 0.95]

alarm	ERRLOWOUTPUT	Moderate	[0.05, 0.95]
child	Disease	N/A	[0.2, 0.3, 0.25, 0.15, 0.05, 0.05]
child	Disease	Little	[0.2, 0.3, 0.25, 0.15, 0.05, 0.05]
child	Disease	Moderate	[0.03, 0.334, 0.295, 0.23, 0.05, 0.05]
hailfinder	SatContMoist	N/A	[0.15, 0.2, 0.4, 0.25]
hailfinder	LLIW	N/A	[0.12, 0.32, 0.38, 0.18]
hailfinder	SatContMoist	Little	[0.15, 0.2, 0.4, 0.25]
hailfinder	LLIW	Little	[0.12, 0.32, 0.38, 0.18]
hailfinder	SatContMoist	Moderate	[0.15, 0.2, 0.4, 0.25]
hailfinder	LLIW	Moderate	[0.12, 0.32, 0.38, 0.18]
insurance	MedCost	N/A	[1.0, 0.0, 0.0, 0.0]
insurance	ILiCost	N/A	[1.0, 0.0, 0.0, 0.0]
insurance	PropCost	N/A	[0.0, 0.0, 0.0, 1.0]
insurance	MedCost	Little	[0.95, 0.03, 0.01, 0.01]
insurance	ILiCost	Little	[0.999, 0.000998, 1e-06, 1e-06]
insurance	PropCost	Little	[0.0, 0.0, 0.95, 0.05]
insurance	MedCost	Moderate	[1.0, 0.0, 0.0, 0.0]
insurance	ILiCost	Moderate	[0.8, 0.1, 0.06, 0.04]
insurance	PropCost	Moderate	[0.0, 0.0, 0.6, 0.4]
Win95pts	Problem1	N/A	[1.0, 0.0]
Win95pts	Problem2	N/A	[1.0, 0.0]
Win95pts	Problem3	N/A	[1.0, 0.0]
Win95pts	Problem4	N/A	[0.0, 1.0]
Win95pts	Problem5	N/A	[0.0, 1.0]
Win95pts	Problem6	N/A	[0.0, 1.0]
Win95pts	Problem1	Problem1	[1.0, 0.0]
Win95pts	Problem2	Problem1	[1.0, 0.0]
Win95pts	Problem3	Problem1	[1.0, 0.0]
Win95pts	Problem4	Problem1	[0.0, 1.0]
Win95pts	Problem5	Problem1	[1.0, 0.0]
Win95pts	Problem6	Problem1	[1.0, 0.0]
Win95pts	Problem1	Problem2	[1.0, 0.0]
Win95pts	Problem2	Problem2	[1.0, 0.0]
Win95pts	Problem3	Problem2	[0.0, 1.0]
Win95pts	Problem4	Problem2	[0.0, 1.0]
Win95pts	Problem5	Problem2	[0.0, 1.0]
Win95pts	Problem6	Problem2	[0.0, 1.0]
Win95pts	Problem1	Problem3	[1.0, 0.0]
Win95pts	Problem2	Problem3	[1.0, 0.0]
Win95pts	Problem3	Problem3	[0.0, 1.0]
Win95pts	Problem4	Problem3	[1.0, 0.0]
Win95pts	Problem5	Problem3	[0.0, 1.0]
Win95pts	Problem6	Problem3	[1.0, 0.0]
Win95pts	Problem1	Problem4	[1.0, 0.0]
Win95pts	Problem2	Problem4	[0.0, 1.0]

Win95pts	Problem3	Problem4	[0.0, 1.0]
Win95pts	Problem4	Problem4	[1.0, 0.0]
Win95pts	Problem5	Problem4	[0.0, 1.0]
Win95pts	Problem6	Problem4	[0.0, 1.0]
Win95pts	Problem1	Problem5	[1.0, 0.0]
Win95pts	Problem2	Problem5	[0.0, 1.0]
Win95pts	Problem3	Problem5	[1.0, 0.0]
Win95pts	Problem4	Problem5	[1.0, 0.0]
Win95pts	Problem5	Problem5	[1.0, 0.0]
Win95pts	Problem6	Problem5	[1.0, 0.0]
Win95pts	Problem1	Problem6	[1.0, 0.0]
Win95pts	Problem2	Problem6	[1.0, 0.0]
Win95pts	Problem3	Problem6	[0.0, 1.0]
Win95pts	Problem4	Problem6	[0.0, 1.0]
Win95pts	Problem5	Problem6	[0.0, 1.0]
Win95pts	Problem6	Problem6	[1.0, 0.0]

6. Analysis and Discussion

The exact inference algorithm had a problem with it, as seen by it returning the same results over the child variable without regard for the amount of given evidence. This may be due to the pre-processing step in which the book's algorithm was deviated from. The exact inference algorithm was also incredibly slow, being run for over 12 hours without returning results for the majority of the studies.

Gibbs sampling consistently returned similar marginal distributions. Many of these marginal distributions were consistent with the marginal distributions given in the BIF files. We went through and made sure that the distributions were being processed, which they were. The Gibbs sampling algorithm ran very quickly, in a matter of a few seconds.

7. Conclusion

Both algorithms solve the problem of determining a variable's marginal distribution based on a Bayesian network and given evidence. They both operate in different ways, where exact inference when implemented correctly returns an optimal solution, and Gibbs sampling returns a close to optimal solution in a much faster time period.

The main difference that we found between the two algorithms was the amount of time it took each to run. Exact Inference was too slow to be actually effective in determining probabilities. Approximation methods exist to make Exact Inference faster, and methods exist in our languages to speed up Exact Inference, given more time these could have been implemented in order to generate results for the algorithm. Gibbs Sampling although not optimal generated similar solutions with much lower run times. In a real world setting, Gibbs Sampling would be chosen as the more usable algorithm.

References

[Kah62] A. B. Kahn. "Topological Sorting of Large Networks". In: *Commun. ACM* 5.11 (Nov. 1962), pp. 558–562. ISSN: 0001-0782. DOI: 10.1145/368996.369025. URL: https://doi.org/10.1145/368996.369025.

[Coo90] Gregory F. Cooper. "The computational complexity of probabilistic inference using bayesian belief networks". In: *Artificial Intelligence* 42.2 (1990), pp. 393–405. ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(90)90060-D. URL: https://www.sciencedirect.com/science/article/pii/000437029090060D.

[RN22] Stuart J. Russell and Peter Norvig. "Chapter 13.4 Approximate Inference for Bayesian Networks". In: *Artificial Intelligence: A modern approach*. Fourth. Pearson Education Limited, 2022.