

On the Convergence Error in Loopy Propagation

Janneke H. Bolt Linda C. van der Gaag

Institute of Information and Computing Sciences, Utrecht University
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands
{`janneke,linda`}@`cs.uu.nl`

Abstract

Reasoning with a Bayesian network amounts to computing probability distributions for the network's variables. Pearl's propagation algorithm provides for efficient reasoning with singly connected networks. When applied to a multiply connected network, the algorithm no longer yields exact probabilities, yet has been reported to result in good approximations. In this paper we provide some theoretical background to the algorithm's performance on multiply connected networks. We indicate the two types of error that can arise in the probabilities computed by the algorithm and analyse the different factors that govern one of these errors.

1 Introduction

Bayesian networks [1] by now are generally accepted as powerful models for capturing the knowledge of complex problem domains along with the uncertainties involved. A Bayesian network consists of a directed acyclic graph in which each node represents a statistical variable and each arc expresses a probabilistic relationship between the connected variables; to capture the strengths of these relationships, each variable has associated conditional probability distributions that describe the effect of all possible combinations of values for its parents on the probabilities of its values. A Bayesian network uniquely defines a joint probability distribution and as such provides for computing any probability of interest over its variables.

Reasoning with a Bayesian network amounts to computing (posterior) probability distributions for the variables involved. For networks without any topological restrictions, this reasoning task is known to be NP-hard [2]. For networks with specific restricted topologies, however, efficient algorithms are available, such as Pearl's propagation algorithm for singly connected networks. Also the task of computing approximate probabilities with guaranteed error bounds is NP-hard in general [3]. Although their results are not guaranteed to lie within given error bounds, various approximation algorithms are available that yield good results on many real-life networks. One of these algorithms is the *loopy-propagation algorithm*. The basic idea of this algorithm is to apply Pearl's propagation algorithm to a Bayesian network regardless of its topological structure. While the algorithm

results in exact probability distributions for a singly connected network, it yields approximate probabilities for the variables of a multiply connected network. Good approximation performance has been reported with the algorithm [4].

Many researchers have addressed the performance of the loopy-propagation algorithm [5, 6]. Weiss and his co-workers more specifically investigated its performance by studying the application of an equivalent algorithm on pairwise Markov networks; their use of Markov networks for this purpose was motivated by the relatively easier analysis of these networks and justified by the observation that any Bayesian network can be converted into a pairwise Markov network. They derived an analytical relationship between the exact probabilities and the computed approximate probabilities for the loop nodes in a network including a single loop.

In this paper we study the application of the loopy-propagation algorithm on Bayesian networks directly, and thereby enhance the insight into its performance. We argue that two types of error can arise in the approximate probabilities yielded by the algorithm, which we term the *cycling error* and the *convergence error*. A cycling error arises when messages are being passed on within a loop repetitively and old information is mistaken for new by the variables involved. A convergence error arises when messages that originate from dependent variables within a loop are combined as if they were independent. As a first step towards its understanding, we analyse the various factors that govern the convergence error in binary networks with loops of restricted topology. We argue that these factors pertain to the degree of dependence between the variables from which the combined messages originate and the extent to which this dependence can affect the computed probabilities.

The paper is organised as follows. In Section 2, we provide some preliminaries on Bayesian networks; in addition, we detail Pearl’s propagation algorithm for singly connected networks. In Section 3, we indicate the two types of error that can arise when Pearl’s algorithm is applied to a multiply connected network. In Section 4, we study the different factors that govern the convergence error. The paper ends with our conclusions and directions for further research in Section 5.

2 Preliminaries

In Section 2.1 we provide some preliminaries on Bayesian networks; in Section 2.2 we review Pearl’s algorithm for reasoning with singly connected networks.

2.1 Bayesian Networks

A *Bayesian network* is a graphical model of a joint probability distribution \Pr on a set of statistical variables. For ease of exposition we assume all variables to be binary, taking one of the values *true* and *false*; we will write a for $A = \textit{true}$ and \bar{a} for $A = \textit{false}$. Each variable is represented by a node in a directed acyclic graph; from now on, we will use the terms node and variable interchangeably. The probabilistic relationships between the variables are captured by the digraph’s set of arcs. The absence of an arc $A \rightarrow B$ between the variables A and B indicates that there is no direct influence between them. If all trails between A and B are

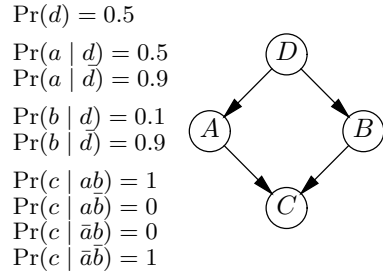


Figure 1: An example Bayesian network.

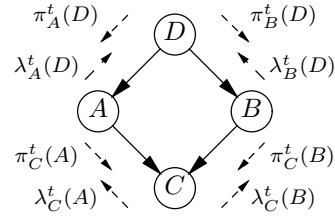


Figure 2: The example network with the messages of Pearl's propagation algorithm.

blocked by evidence, moreover, then there also is no indirect influence of A on B or vice versa. We say that a trail between A and B is blocked by the available evidence if it includes either an observed variable with at least one outgoing arc, or an unobserved variable with two incoming arcs and no observed descendants. If there are no direct or indirect influences between A and B , then the variables are taken to be conditionally independent given the evidence. Associated with the digraph are numerical quantities from the modelled distribution: for each variable A , a set of conditional probability distributions $\Pr(A | \pi(A))$ is specified, where $\pi(A)$ denotes the set of parents of A in the digraph. As an example, Figure 1 depicts a small Bayesian network. If no observations have been entered as yet, its variables A and B are dependent along the trail $A \leftarrow D \rightarrow B$. The two variables become independent as soon as an observation for the variable D is entered. If observations are entered for both D and C , then the variables A and B may be dependent again, this time along the trail $A \rightarrow C \leftarrow B$.

In the sequel, we distinguish between singly connected and multiply connected Bayesian networks. A network is *singly connected* if there is at most one trail between any two variables in its digraph. If there are multiple trails between variables, then the network is *multiply connected*. A multiply connected Bayesian network includes one or more loops, that is, one or more cycles in its underlying undirected graph. A node that has two or more incoming arcs on the loop will be called a *convergence node*. The Bayesian network from Figure 1 is an example of a multiply connected network. The trail $A \leftarrow D \rightarrow B \rightarrow C \leftarrow A$ constitutes a loop in the network's digraph; node C is the only convergence node in this loop.

2.2 Pearl's Propagation Algorithm

Pearl's propagation algorithm was designed for probabilistic reasoning with singly connected Bayesian networks [1]. The algorithm is based upon the idea of message passing between neighbouring variables. In the algorithm, each node X is looked upon as an autonomous object that applies a limited number of simple computation rules. Using these rules, the node can compute its probability distribution $\Pr(X | \mathbf{e})$ given the available evidence \mathbf{e} , from the messages it receives from its neighbours. Node X in turn computes the messages it has to send to its neighbours

to enable them to compute their probability distributions. The computation rules are applied by the various nodes in parallel. In each iteration of the algorithm, a node takes the messages it received in the previous time step for its input. The rule used by node X for computing the probability distribution $\Pr(X | \mathbf{e})$ now is

$$\Pr^t(x | \mathbf{e}) = \alpha \cdot \lambda^t(x) \cdot \pi^t(x)$$

where $\lambda^t(x)$ is computed from the *diagnostic messages* $\lambda_{Y_j}^t(x)$ it receives from each of its children Y_j :

$$\lambda^t(x) = \prod_j \lambda_{Y_j}^t(x)$$

and $\pi^t(x)$ is computed from the *causal messages* $\pi_X^t(U_i)$ it receives from each of its parents U_i :

$$\pi^t(x) = \sum_{U_1, \dots, U_n} \left(\Pr(x | U_1, \dots, U_n) \cdot \prod_i \pi_X^t(U_i) \right)$$

The rule for computing the diagnostic message to be sent to its parent U_i is

$$\lambda_X^{t+1}(u_i) = \sum_x \lambda^t(x) \cdot \sum_{U_k: k \neq i} \left(\Pr(x | U_1, \dots, U_n) \cdot \prod_{k \neq i} \pi_X^t(U_k) \right)$$

and the rule for computing the causal message to be sent to its child Y_j is

$$\pi_{Y_j}^{t+1}(x) = \beta \cdot \pi^t(x) \cdot \prod_{k \neq j} \lambda_{Y_k}^t(x)$$

where α and β are normalisation constants. Figure 2 indicates, as an example, the various messages that are sent between the variables of the network from Figure 1. We note that in the prior state of a network all diagnostic messages are equal to 1. An observation for a node X can be entered into the network by multiplying $\lambda^t(x)$ by 1 for the observed value of X and by 0 for the other value(s).

3 Errors in Loopy Propagation

When applied to a singly connected Bayesian network, Pearl's propagation algorithm results in exact probabilities. When applied to a multiply connected network, however, the computed probabilities may include two types of error.

The first type of error arises when messages are being passed on in a loop repetitively and old information is mistaken for new by the variables involved. As an example, we consider again the multiply connected network from Figure 1. After an observation for node C has been entered, the diagnostic message that it sends to node A will consist of information about its own conditional probabilities combined the probability distribution of node B . Node A passes this information on to node D which in turn sends it to B . Node B misinterprets its own information for new and includes it into its probability distribution. Node B then sends a

different message to node C and the updating process is repeated. The error that thus arises will be termed a *cycling error*.

The second type of error originates from the combination of causal messages by the convergence nodes of the loops. A convergence node combines the messages from its parents as if they come from independent sources. In a singly connected network the parents of a node indeed are independent, the parents of a convergence node, however, may be dependent. By assuming independence upon combining the causal messages, a *convergence error* may then be introduced.

4 The Convergence Error

To study the size of the convergence error, we apply the loopy-propagation algorithm to the example Bayesian network from Figure 1. In the prior state of the network, all diagnostic messages equal 1 and the nodes receive informative messages from their parents only. In this state, moreover, the algorithm establishes exact probability distributions for the nodes A , B and D . Node D sends a causal message with its exact prior probability distribution to A and B . These nodes combine the message they receive from D with their own conditional probabilities in a mathematically correct way. They subsequently send their exact prior probability distributions to the convergence node C which combines the two messages to establish its own (approximate) probability distribution. Since node C does not pass on any information to nodes A and B , only a convergence error arises during the propagation. Studying the prior state of the network therefore allows us to investigate the convergence error in isolation.

Upon receiving the exact prior probability distributions of the nodes A and B , node C establishes the following approximate probability:

$$\begin{aligned}\widetilde{\Pr}(c) &= \sum_{A,B} \Pr(c | AB) \cdot \pi_C(A) \cdot \pi_C(B) \\ &= \sum_{A,B} \Pr(c | AB) \cdot \Pr(A) \cdot \Pr(B)\end{aligned}$$

where we use $\widetilde{\Pr}$ to distinguish approximate probabilities from exact ones. The computation rule used for the derivation explicitly builds upon the assumption that the two parents A and B of C are independent. This assumption holds for the singly connected networks for which the algorithm was developed, yet does not hold for multiply connected networks in general. In the network from Figure 1 more specifically, nodes A and B are dependent, thereby violating the assumption underlying the computation rule which in turn gives rise to a convergence error.

To establish the size of the convergence error in the computed probability $\widetilde{\Pr}(c)$, we compare it against the exact probability $\Pr(c)$; the latter probability equals

$$\begin{aligned}
\Pr(c) &= \sum_{A,B} \Pr(c | AB) \cdot \Pr(AB) \\
&= \sum_{A,B,D} \Pr(c | AB) \cdot \Pr(AB | D) \cdot \Pr(D) \\
&= \sum_{A,B,D} \Pr(c | AB) \cdot \Pr(A | D) \cdot \Pr(B | D) \cdot \Pr(D)
\end{aligned}$$

Note that the derivation builds upon the observation that the parents A and B of C are independent given D . The difference between the exact and approximate probabilities now is

$$\Pr(c) - \widetilde{\Pr}(c) = w \cdot x \cdot y \cdot z$$

where

$$\begin{aligned}
w &= \Pr(c | ab) - \Pr(c | a\bar{b}) - \Pr(c | \bar{a}b) + \Pr(c | \bar{a}\bar{b}) \\
x &= \Pr(a | d) - \Pr(a | \bar{d}) \\
y &= \Pr(b | d) - \Pr(b | \bar{d}) \\
z &= \Pr(d) - \Pr(d)^2
\end{aligned}$$

We observe that the size of the convergence error is governed by four factors. These factors are illustrated in Figure 3; for the construction of the figure we used the probabilities from Figure 1. The surface shown in Figure 3 captures the approximate probability $\widetilde{\Pr}(c)$ computed for the convergence node C as a function of the probabilities $\Pr(a)$ and $\Pr(b)$; for any particular $\Pr(a)$ and $\Pr(b)$, therefore, the corresponding $\widetilde{\Pr}(c)$ is a point on this surface. The figure further shows a line segment that has its two endpoints on the depicted surface. This line segment expresses the exact probability $\Pr(c)$ as a function of $\Pr(d)$; for a particular $\Pr(d)$, resulting in a particular $\Pr(a)$ and $\Pr(b)$, therefore, the corresponding $\Pr(c)$ is a point on the line segment. Note that while the surface depicts the probability of c under the assumption of independence of A and B , the line segment takes the dependence between these two nodes into consideration. The convergence error now equals the distance between the point on the line segment that matches $\Pr(d)$ and its orthogonal projection on the surface; for $\Pr(d) = 0.5$, the difference between $\widetilde{\Pr}(c)$ and $\Pr(c)$ is indicated by the vertical dotted line segment.

The impact of the various factors on the convergence error now is as follows:

- The factor w captures the curvature of the surface. The more curved the surface is, the larger the distance between a point on the line segment and its projection on the surface will be. The larger the factor w , therefore, the larger the convergence error. The factor ranges between -2 and 2 . For our example network, we have a maximal curvature of the surface and $w = 2$.
- The distance between a point on the line segment and its projection on the surface is maximal for the segment's midpoint and minimal for its endpoints. The impact of the location of this point on the convergence error is reflected by the factor z . The value of z ranges between 0 and 0.25 . For our example

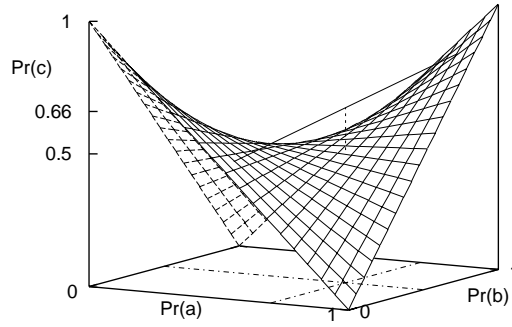


Figure 3: The probability of c as a function of $\Pr(a)$ and $\Pr(b)$, assuming independence of the parents A and B of C (surface), and as a function of $\Pr(d)$, assuming independence of A and B given D (line segment).

network, with $\Pr(d) = 0.5$, the exact probability $\Pr(c)$ is located at the midpoint of the line segment and z is maximal. Would $\Pr(d)$ have been equal to 0 or to 1, then the probability $\Pr(c)$ would have been located at an endpoint of the line segment and, hence, would have lied on the depicted surface. Note that with these values of $\Pr(d)$, the nodes A and B would have been independent and exact probabilities for C would indeed have been found.

- The distance between a point on the line segment and its projection on the surface further depends on the location of the line segment. The distance is zero if the projection of the line segment on the $\Pr(a), \Pr(b)$ -plane of the figure is orthogonal to one of the axes; it is maximal if the projection equals one of the plane's diagonals. Moreover, the distance is zero if the two endpoints of the line segment coincide; it is maximal if these points have maximal distance. The impact of the location of the line segment on the convergence error is expressed by the factors x and y . These factors range between 1 and -1 ; in our example network, we have $x = -0.4$ and $y = -0.8$.

Informally speaking, the factors x , y and z with each other capture the degree of dependence between the nodes A and B along the trail through node D and the factor w indicates to which extent this dependence can affect the computed probabilities. From the above considerations, we note that the error ranges between -0.5 and 0.5 . We now conclude our analysis by establishing the size of the convergence error for node C in the example network. For the four factors that govern the error, we found that $w = 2$, $x = -0.4$, $y = -0.8$ and $z = 0.25$, from which we establish a convergence error of $0.25 \cdot -0.4 \cdot -0.8 \cdot 2 = 0.16$. Using the loopy-propagation algorithm, we compute from the network the approximate probability $\widehat{\Pr}(c) = 0.5$. Note that our closed formula for the convergence error enables us to establish the exact probability to be $\Pr(c) = \widehat{\Pr}(c) + 0.16 = 0.66$.

So far, we analysed the size of the convergence error just for the example network from Figure 1. Our analysis readily extends to binary networks in which each convergence node has just two loop nodes among its parents. While for our

example network we could establish the various factors that govern the error's size directly from the network's specification, however, this may no longer be possible for networks of more complex topology.

5 Conclusions

We investigated the errors that arise when Pearl's propagation algorithm for reasoning with singly connected networks is applied to a multiply connected Bayesian network. We identified the two types of error that are introduced by the algorithm in the computed probabilities, which we termed the cycling error and the convergence error. For binary networks with loops of restricted topology, moreover, we detailed the various factors that govern the size of the convergence error. This type of error arises when a convergence node in a loop combines the messages it receives from its two parents as if originating from independent sources. The size of the error is subject, among other factors, to the degree of dependence between the two parents. Informally speaking, the less dependent these two nodes are, the smaller the convergence error will be. By studying the application of the algorithm directly on Bayesian networks, we enhanced the insight into its performance, compared to previous research that used pairwise Markov networks for this purpose.

We studied the convergence error only in binary networks in which each convergence node has just two loop nodes among its parents. When a convergence node has more than two incoming arcs from loop nodes, the formula expressing the convergence error will be much more complicated. We expect, however, that also the more general expression will bear information about the degree of dependence among the parents involved. Also, we studied the size of the convergence error only in the prior state of a network. When observations are entered into the network, the convergence error will change in size. We hope to report new results from our further analysis of these issues in the near future.

Acknowledgement. This research was partly supported by the Netherlands Organisation for Scientific Research (NWO).

References

- [1] J. Pearl (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Palo Alto.
- [2] G.F. Cooper (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, **42**, 393 – 405.
- [3] P. Dagum, M. Luby (1993). Approximate inference in Bayesian networks is NP hard. *Artificial Intelligence*, **60**, 141 – 153.
- [4] K. Murphy, Y. Weiss, M. Jordan (1999). Loopy belief propagation for approximate inference: an empirical study. In: K.B. Laskey, H. Prade (editors). *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers, San Francisco, 467 – 475.
- [5] Y. Weiss (2000). Correctness of local probability propagation in graphical models with loops. *Neural Computation*, **12**, 1 – 41.
- [6] Y. Weiss, W.T. Freeman (2001). Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, **13**, 2173 – 2200.