Towards Improved Bayesian Fusion through Run-time Model Analysis

Jan Nunnink Gregor Pavlin
IAS Group, Informatics Institute,
Faculty of Science
University of Amsterdam
jnunnink@science.uva.nl

Abstract—This paper considers the accuracy of state estimation based on classification using Bayesian networks. It presents a method to localize network fragments that (i) are in a particular (rare) case responsible for a potential misclassification, or (ii) contain modeling errors that consistently cause misclassifications, even in common cases. We derive an algorithm that, within such fragments, can localize the probable cause of the misclassification. The approach is based on monitoring the Bayesian network’s ‘behavior’ at runtime, specifically the correlation among sets of evidence. We suggest several applications for the algorithm’s output, such as repairing or mitigating the effects of errors, or deactivating faulty information sources.

Keywords: Bayesian networks, model analysis.

I. INTRODUCTION

Contemporary decision making processes require accurate and quick situation assessment, such as detection of toxic gases, disease outbreaks, fires, etc. In such settings, critical hidden events must be inferred through interpretation (i.e. fusion) of large quantities of uncertain and very heterogeneous information. The information can be obtained via sensor networks, human reports, databases, etc. Interpretation of heterogeneous types of information requires adequate domain models which provide a mapping between the heterogeneous observations and hypotheses of interest. Domains are often complex, which means that models inevitably are abstractions associated with significant uncertainties.

It turns out that discrete Bayesian networks (BNs) are a suitable framework in such a context. BNs are powerful probabilistic models for reasoning about uncertainty, and support systematic fusion of heterogeneous and noisy information. They represent relations between very heterogeneous types of information in a rigorous and efficient, yet intuitive manner. Efficient inference algorithms exist that can be used for the actual fusion process. BNs are also often used as the basis for classification in decision making, or for state estimation in a filtering context.

The success of the Bayesian network approach depends on the quality of the model. Errors can be introduced in a BN through a variety of reasons. In this context, we emphasize the difference between the generalization accuracy and the classification accuracy. In general, classification is based on models that are generalizations over many different situations. BNs capture such generalizations through conditional probability distributions over related events. However, accurate generalizations do not guarantee accurate classification in a particular case. In a rare situation, a set of observations could result in erroneous classification, even if the model precisely described the true distributions (i.e. the distributions used by ‘nature’).

For example, consider a domain in which 90% of fires cause smoke. While it is common to have fire and smoke coexisting in a case, rare cases will occur where we have a smokeless fire. Observing the absence of smoke would in such a rare case decrease our belief in the presence of fire, leading our belief away from the truth, even if the used BN were a perfect generalization.

We can assume that every variable in a model has a hidden true state, which we are trying to find through fusion of information. Each fragment of a BN either supports this true state, by increasing its likelihood, or not. We call a fragment that supports the true state adequate. An error or rare case can cause a fragment to be inadequate.

In certain applications, state estimations are critical to the further course of events. Therefore, it seems prudent to develop methods that, at a relatively low cost, can monitor a BN during run-time, and detect errors. In this paper we present such a method, based on the BN properties presented in [1].

Inadequacies influence the inference process and subsequent classification. This is reflected in the way the model ‘behaves’ under different circumstances. We can monitor a BN and from its behavior draw conclusions about the existence of inadequate parts. This is based on the following principle: A given classification variable splits the network in several conditional independent fragments. These fragments can be seen as different experts giving independent ‘votes’ about the state of that variable. The degree to which they ‘agree’ on a state is a measure for the accuracy of the classification.

This idea was also explored in [1]. The data conflict approach by Jensen et al. [2] (as well as [3] and [4]) uses a roughly similar principle. It is based on the assumption that given an adequate model, all observations should be correlated and hence the joint probability of all evidence should be greater than the product of the individual evidence probabilities. The advantage of our method is that, by making a few plausible assumptions, we can give a lower bound on its effectiveness, and show that this lower bound has asymptotic properties given the network structure.
II. State Estimation

Given a domain of variables \( \mathcal{V} = \{V_1, \ldots, V_n\} \), a Bayesian network \( BN \) defines a factorization of the joint probability distribution (JPD) over the domain \([5], [6]\). This factorization is represented with a directed acyclic graph (DAG) over \( \mathcal{V} \). Each variable corresponds to exactly one node in the graph, and arcs between nodes represent direct probabilistic dependencies. The factorization is then as follows:

\[
p(V) = \prod_{V_i \in \mathcal{V}} p(V_i | \pi(V_i)),
\]

where \( \pi(V_i) \) are the parent nodes of \( V_i \) in the graph. The factors, \( p(V_i | \pi(V_i)) \), are conditional probability distributions (CPDs), describing the probabilistic relations between variables. In many applications, the graph represents the causal structure of the domain, and the conditional probabilities encode the causal strength.

Each variable has a finite number of discrete states (or values), we will denote by lowercase letters. Furthermore, some of the variables in the domain might be ‘observed’, meaning that we know their state. These observation are called evidence, and we denote it by \( \mathcal{E} \). In the case of state estimation, we will often denote the variable whose state we are trying to estimate by \( H \). The (hidden) true state of \( H \) in a particular case is denoted by \( h^* \).

State estimation of some \( H \) is done by computing the probability distribution over \( H \) given all evidence, and taking the most probable state. Formally, \( H \) is classified as \( h_i \) iff

\[
h_i = \arg \max_{h_j} p(H = h_j, \mathcal{E}).
\]

\( p(H = h_j, \mathcal{E}) \) can be obtained using an appropriate inference algorithm.

A. Fragments and Factors

In this section and the next we summarize some of the results from [1] and [7]. See those for a more thorough discussion.

For the analysis in the next sections we require a notion of dependence between different parts of a graph given the classification node. We can use d-separation [6] to identify fragments of the graph which are conditionally independent given the classification node.

Definition 1: Given a graph \( G \) and classification node \( H \), we can identify sets of nodes \( \mathcal{V}_k \), where every set is d-separated from the other sets given \( H \). A fragment \( \mathcal{F}_k^H \) is defined as the subgraph of \( G \) consisting of the nodes in \( \mathcal{V}_k \) plus \( H \). Node \( H \) is called the root of fragment \( \mathcal{F}_k^H \). The number of fragments rooted in \( H \) is called the branching factor.

See Figure 1 for an example. Given node \( H \), 3 fragments can be identified, containing (i) nodes \( \{A, B, C, E, H\} \), (ii) nodes \( \{F, H\} \), and (iii) nodes \( \{G, H\} \).

This fragmentation has the useful property that \( p(H = h_i, \mathcal{E}) \), which is used for classification in (1), can be factorized such that each factor corresponds one-on-one with a fragment.

By splitting the sum and product and regrouping them per fragment, we can write:

\[
p(H = h_i, \mathcal{E}) = \sum_{\mathcal{V}\setminus H} \prod_{V_i \in \mathcal{V}} p(V_i | \pi(V_i)) \]

\[
= p(h_i | \pi(H)) \sum_{V_i \in \mathcal{V}} \prod_{V_i \in \mathcal{V}} p(V_i | \pi(V_i)) \cdot \phi_i(h_i)
\]

\[
\cdots
\]

\[
\sum_{\mathcal{V}_k \setminus E} \prod_{V_i \in \mathcal{V}_k} p(V_i | \pi(V_i)) \cdot \phi_k(h_i)
\]

By splitting the sum and product and regrouping them per fragment, we can write:

\[
p(H = h_i, \mathcal{E}) = \sum_{\mathcal{V}\setminus H} \prod_{V_i \in \mathcal{V}} p(V_i | \pi(V_i)) \]

\[
= p(h_i | \pi(H)) \sum_{V_i \in \mathcal{V}} \prod_{V_i \in \mathcal{V}} p(V_i | \pi(V_i)) \cdot \phi_i(h_i)
\]

\[
\cdots
\]

\[
\sum_{\mathcal{V}_k \setminus E} \prod_{V_i \in \mathcal{V}_k} p(V_i | \pi(V_i)) \cdot \phi_k(h_i)
\]

By splitting the sum and product and regrouping them per fragment, we can write:

\[
p(H = h_i, \mathcal{E}) = \sum_{\mathcal{V}\setminus H} \prod_{V_i \in \mathcal{V}} p(V_i | \pi(V_i)) \]

\[
= p(h_i | \pi(H)) \sum_{V_i \in \mathcal{V}} \prod_{V_i \in \mathcal{V}} p(V_i | \pi(V_i)) \cdot \phi_i(h_i)
\]

\[
\cdots
\]

\[
\sum_{\mathcal{V}_k \setminus E} \prod_{V_i \in \mathcal{V}_k} p(V_i | \pi(V_i)) \cdot \phi_k(h_i)
\]

We can identify factors \( \phi_k(h_i) \) whose product is the joint probability for each \( h_i \) with the evidence. \( \phi_k(h_i) \) denotes the value of factor \( \phi_k \) for \( h_i \). The d-separation between the fragments directly implies that the factors in (2) are mutually independent, given classification variable \( H \).

B. Accuracy and Adequacy

Recall that \( h^* \) denotes the true (hidden) state of variable \( H \) and that we classify \( H \) according to (1). We can say that a classification is accurate if

\[
h^* = \arg \max_{h_i} p(H = h_i, \mathcal{E}).
\]

In other words, the true state should have the greatest probability.

If we plug factorization (2) into (3), we can extend accuracy to fragments. Recall that \( \phi_k(h_i) \) is the value of factor \( \phi_k \) for state \( h_i \). Since factors are mutually independent, we can say that factor \( \phi_k \) and corresponding fragment \( \mathcal{F}_k^H \) support an accurate classification if

\[
h^* = \arg \max_{h_i} \phi_k(h_i).
\]

Definition 2: If a fragment supports an accurate classification in a given situation, we call this fragment and its associated factor adequate.

Furthermore, we say that factor \( \phi_k \) reinforces state \( \arg \max_{h_i} \phi_k(h_i) \). The intuition behind this is that if a factor gives its highest value to the true state, it will help the true state to become the most probable state, leading to an accurate classification. Note that computing \( \arg \max_{h_i} \phi_k(h_i) \) for every variable in the network takes no additional time, because the factor values are a byproduct of most inference algorithms, which are based on passing factor-messages between nodes.

Our goal is to find out which fragments are adequate. According to (4), if we know the true state, we can determine which fragments of a Bayesian network are adequate. Since
we do not have this knowledge, however, a different approach is required. In the next section we will present a method that can indicate whether a fragment is adequate. In Section IV, an algorithm is presented that can determine which CPT is likely to be the cause of an inadequate fragment.

III. Adequacy Estimation

A. Factor Consistency

Since the true state of a hidden variable is unknown, it is impossible to directly determine whether or not (4) is satisfied. We can however use the following definition which describes the relation between multiple factors: Given any variable \( H \), a set of factors \( \Phi_H \) is consistent iff

\[
\forall \phi_i, \phi_l \in \Phi_H \quad \arg\max_{h_i} \phi_k(h_i) = \arg\max_{h_i} \phi_l(h_i).
\]

In other words, factors are consistent if they reinforce the same state of \( H \). Given that there can be only one true state \( h^* \) at any moment, it is clear that if each factor in \( \Phi_H \) satisfies (4), then that set must be consistent.

Consequently, if a set of factors is not consistent, then some of the factors in that set do not satisfy (4). In that case we should determine which of the factors in an inconsistent set violate (4).

B. Consistency Measure

First, we require a measure for the degree of consistency of a factor with respect to the observed factor reinforcements of all other factors in a set \( \Phi_H \). Consider a node \( H \) with states \( \{h_1, \ldots, h_m\} \) and associated reinforcement counters \( N = \{n_1, \ldots, n_m\} \). A counter \( n_i \) denotes the number of factors in \( \Phi_H \) that reinforce state \( h_i \).

Suppose that factor \( \phi_k \) reinforces state \( h_i \). The consistency measure for \( \phi_k \in \Phi_H \) is defined as:

\[
C_H(\phi_k) = n_i - \max_{j \neq i} n_j,
\]

In other words, the consistency measure for a factor \( \phi_k \) is equal to the number of factors ‘agreeing’ with \( \phi_k \) (including \( \phi_k \) itself), minus the maximum number of reinforcements any other state \( h_j \) of \( H \) got.

C. Fragment Adequacy

The consistency measure can be used to determine the fragment adequacy. The key idea is that a fragment associated with a high consistency (i.e. there exist many other fragments which ‘agree’ with it) is likely to be adequate. This is true if the probability \( p_\phi \) that a fragment is adequate in a given situation is greater than 50%. A thorough rationale defending this assumption would not fit in this paper, but it can be found in [1] and [7]. In this paper we just illustrate why this assumption is plausible.

In general, \( p_\phi \) depends on the true distributions (as exist in the domain one tries to model) over variables, as well as simple relations between the true distributions and the CPT parameters. For BNs with tree topologies we can show that \( p_\phi > 0.5 \) if (i) the true conditional probabilities for any pair of dependent variables from a fragment satisfy simple greater/smaller-than relations that are also captured by the corresponding CPT parameters, and (ii) the counters needed for determining the consistency measures are computed with the help of the reinforcement propagation algorithm [1] (A simplified fusion algorithm which has asymptotic properties w.r.t. the classification accuracy). These relations are especially simple in trees consisting of binary variables. In such a case, for any fragment \( p_\phi > 0.5 \) if both:

- The true conditional distribution \( \hat{p}(E|C) \) between any pair of dependent variables \( C \) and \( E \) are such that either \( \hat{p}(e|c) > 0.5 \wedge \hat{p}(\bar{e}|ar{c}) > 0.5 \) or \( \hat{p}(\bar{e}|c) > 0.5 \wedge \hat{p}(e|ar{c}) > 0.5 \)
- The corresponding CPT parameters in the model satisfy the same relations.

It is plausible to assume that designers or learning algorithms can easily identify coarse relations between the conditional distributions and construct CPTs whose parameters capture such relations correctly.

Given \( p_\phi > 0.5 \), we can express bounds on the probability that a particular consistency measure corresponds to an adequate fragment.

\( N \) is the total number of factors. Suppose \( \phi \) is the factor under consideration, and it reinforced state \( h_i \), in other words \( \arg\max_{h_i} \phi(h_j) = h_i \). The conditional probability that this \( h_i \) equals the true state \( h^* \), given only the counters \( N \), can be expressed as

\[
p(h_i = h^*|N) = \frac{p^{|N|}_{\phi}(1 - p_\phi)^{N-n_i}}{\sum_j p^{|N|}_{\phi}(1 - p_\phi)^{N-n_j}}.
\]

The numerator consists of the probability of an adequate fragment to the power of the number of fragments agreeing with \( \phi \), times the probability that a fragment is inadequate to the power of the number of reinforcements not supporting \( h_i \). The denominator normalizes the distribution.

We want to determine exactly for which degree of consistency \( C_H \) this conditional probability is greater (or smaller) than 0.5. See [8] for the complete derivation; here we just give the results.

We derive the following implications:

\[
C_H(\phi) < 0 \Rightarrow p(\arg\max_{h_j} \phi(h_j) = h^*|N) < 0.5
\]

\[
C_H(\phi) > \frac{\log(n-1)}{\log p_\phi/(1-p_\phi)} \Rightarrow p(\arg\max_{h_j} \phi(h_j) = h^*|N) > 0.5
\]

where \( m \) is the number of states of \( H \). These implications give the probability of an accurate factor reinforcement, given its consistency measure. This allows us to use an easily observable quantity (the counters \( n_i \), in order to derive a bound on the probability that a particular fragment is adequate. Thus, if a factor has a negative consistency measure, the corresponding fragment probably contains an error.
Implication (6) is not trivial to interpret, since the condition depends on the unknown \( p_{\phi_1}(1 - p_{\phi_2}) \). It turns out, however, that the condition is quite insensitive to its exact value. Therefore, we can just choose a ‘safe’ value for \( p_{\phi_1}(1 - p_{\phi_2}) \), such as 2.

**IV. Fault Localization Algorithm**

Using the results from the previous section, we can identify fragments that are probably inadequate. In addition, whether a fragment in a given situation is adequate or not depends on the CPTs associated with its nodes. In this context we define an adequate CPT as follows:

**Definition 3 (Adequate CPT):** A CPT \( p(E \vert C) \) is adequate in a given case if the following is true: if one variable were instantiated to its true state, then the belief propagation based on this CPT would reinforce the true state of the other variable. In other words, an adequate CPT supports accurate classification in a given situation. If all CPTs from a fragment were adequate in a given situation, then also the fragment itself would be adequate. This is usually not the case, however. Whether a factor \( \phi_i \) is adequate depends on which CPTs from the corresponding fragment are inadequate. Obviously, the higher the probability that any CPT from a fragment is adequate, the higher the probability that the fragment is adequate as well.

The next step is to find which CPT in a fragment is responsible in the case of an inadequacy. Our method is based on a comparison of the fragment adequacy at different nodes in the graph. This comparison gives an indication of the adequacy of the CPT between the nodes.

Consider a network section consisting of two adjacent nodes, \( X \) and \( Y \) (see Figure 2). First we consider one particular fragment \( F_k^x \) rooted in \( X \), and its associated factor \( \phi_k \) (see Figure 2b). We can obtain the consistency measure \( C_Y(\phi_k) \) at node \( Y \). This measure combined with (5) and (6) indicates whether fragment \( F_k^x \) is likely to be adequate.

Let \( F' \) be fragment \( F_k^y \) plus the arc between \( X \) and \( Y \). \( F' \) would be a fragment of \( X \) if we would remove all fragments of \( Y \) except for \( F_k^x \) from the graph (see Figure 2c). Let \( \phi_k' \) be its corresponding factor. The values of this factor are easily computed from the byproducts of the inference algorithm. We can observe the consistency measure \( C_X(\phi_k') \) at node \( X \) for fragment \( F' \) (see Figure 2c). Again, this gives an indication of the adequacy of fragment \( F_k^y \), but this time including arc \( \langle X, Y \rangle \).

These two consistency measures combined indicate the adequacy of the CPT \( p(Y \vert X) \) corresponding to the arc \( \langle X, Y \rangle \). We use the following intuition: If a fragment is adequate up to \( Y \), but the extended fragment is inadequate up to \( X \), then the cause lies with arc \( \langle X, Y \rangle \). All such localization rules are shown in Table I.

In other words, we compare the consistency at two adjacent nodes and classify the arc between the nodes as adequate or inadequate. We can show that this will guarantee that in most cases the (in)adequacy of the CPT corresponding to \( \langle X, Y \rangle \) is correctly determined.

**Proposition 1 (Fault Localization):** Given a network containing a sufficient number of fragments and \( p_{\phi_1} > 0.5 \), fault localization will correctly determine whether a particular CPT \( p(Y \vert X) \) is adequate or inadequate with more than 50% probability.

See [8] for the proof. We can apply fault localization to all non-leaf nodes by running Algorithm 1.

**Algorithm 1: Localization Algorithm**

```plaintext
1 for each node X do
   for each fragment \( F_k^x \) of X do
      for each fragment \( F_k^y \) of Y do
         Compute \( C_X(\phi'_k) \) and \( C_Y(\phi_k) \) for \( F_k^y \); Using (6), (5), and Table I, classify CPT \( p(Y \vert X) \);
      end
      Use majority voting on all classifications of CPT \( p(Y \vert X) \) based on different \( F_k^y \);
   end
end
```

We observe the following property of Algorithm 1: The accuracy of the majority voting at the end of Algorithm 1 improves with higher branching factors. Higher branching factors imply more votes about the state of a CPT and therefore higher expected localization accuracy. This accuracy converges asymptotically to 1 if the branching factors increase.

**A. Inadequacy or Error**

We can distinguish between inadequacies due to rare cases and actual modeling errors by their frequency of occurrence (as they both violate (4)). For this we need to perform fault
localization on a BN for a set of cases. If a certain CPT is identified as inadequate for a large number of cases, then this is an indication that the CPT parameters might not correctly capture the relations as described in Section III-C. Note that the assumption that \( p_0 > 0.5 \) for a particular fragment is still valid if a sufficient majority of the CPTs in that fragment correctly capture those relations. This is a plausible assumption, since we can expect that experts or learning algorithms can easily identify such simple relations for most CPTs.

Alternatively, it might be possible to find model errors by localizing faults on a case from the domain which one knows is not rare. In other words, a case for which we know that the true state of every node is the most likely state given the evidence. This excludes the possibility for inadequacies due to an rare case. Any found inadequacies are then probably caused by modeling errors.

V. EXPERIMENTS

To verify our claims and illustrate some of the properties of algorithm 1 we applied it to several synthetic networks, in which we purposely introduced errors. We also applied it to a real network, which we adapted such that it represents an oversimplification of the problem domain, thus introducing inadequacies due to rare cases.

A. Synthetic Networks

We generated BNs with random CPTs and a tree-shaped graph with fixed branching factor and 4 levels. We initialized all CPTs such that the probability of a CPT being adequate could be controlled. We call this probability \( p_{\text{cpt}} \) and let it take values 1, 0.95, \ldots, 0.4, to see how well the algorithm can cope with different quality levels. Then we generated 1000 data samples for each particular network, making all variables hidden, except for the leaves. We then ran inference on each sample case, applied the Localization Algorithm, and observed its output.

The output was then compared with the ground truth, i.e. which CPTs really were inadequate for the given data case. This ground truth can be obtained from the complete cases, without hidden variables, which were known to us. Given the inadequate CPTs that were present for a given case, we recorded the percentage of inadequate CPTs that the algorithm managed to detect and the percentage of detected inadequacies that turned out to be false positives.

We applied this whole procedure on networks with varying branching factors (but the same general structure). These results are plotted in Figure 3. The three top curves show the detection percentage, and the three bottom curves correspond to false positive percentage. Each of the three lines correspond to different values of \( p_{\text{cpt}} \). The figure confirms that higher branching factors increase the algorithm’s effectiveness, for all values of \( p_{\text{cpt}} \).

Next, we varied the number of states per variable in Figure 4. The different curves correspond to different numbers of states. One observation we can make is that the quality of the algorithm output at a certain point becomes too low to be usable. This is the point where on average almost half of the CPTs are inadequate. However, this is a situation that is very unlikely to occur in real applications.

Figure 4 also shows that the algorithm performs better on networks with more variable states, which can be explained by the fact that in such cases inadequate sample values are spread over more states. For example, suppose that in a certain case a variable is in state 1, but an inadequate fragment supports a different state. If a variable has more states, inaccurate support will be spread among more alternatives. Thus, on average, the difference between the counter of the correct state and the other counters increases, making the correct state still stand out. For example given counters \( \mathcal{N} = \{3, 2\} \), state 1 would have a consistency measure of 1, while for \( \mathcal{N} = \{3, 1, 1\} \) it would be 2. Note that the degree of this spread also influences the quality, as can be seen from the dotted line in Figure 4.

![Figure 3](image-url-3)

Figure 3. The effect of branching factors on a network with 4-state variables, for different values of \( p_{\text{cpt}} \): 0.9 (dash-dotted), 0.7 (solid), 0.5 (dashed). Top curves show percentage found, bottom curves show percentage of false positives.

![Figure 4](image-url-4)

Figure 4. The effect of the number of variable states on a network with branching factor 5, for different values of \( p_{\text{cpt}} \) (horizontal axis). Number of states: 2 (dashed), 3 (solid), 4 (dash-dotted). Top curves show percentage found, bottom curves show percentage of false positives. The dotted line shows the worst case scenario for 3 and 4 states.
A. Localizing Modeling Errors

The localization algorithm can discover CPTs that are inadequate in a particular situation. By applying the localization algorithm to many different samples obtained in different situations, we can localize CPTs which are found to be inadequate in the majority of the samples. Such CPTs represent modeling errors. That is, they do not satisfy the same inequality relations as are present in the true distributions in 'nature'. Such erroneous CPTs can then be repaired manually or through learning.

Errors cannot be avoided if a model is used in changing domains and the learning examples or expertise that generated the model did not capture the characteristics of the new domain. Fault localization can be especially useful in domains which change sufficiently slowly, allowing us to discover local inadequacies and adapt the model gradually to the new domain.

The assumption that a fragment is adequate with at least 0.5 probability was plausible because we assumed that it is easy for model builders to find good CPTs. That is, CPTs which capture the relations in the true distributions correctly and support good classification performance (see [7], [11]). In a given case the inadequate CPTs most likely form a minority. When certain CPTs contain errors they will likely be inadequate in most cases. However, as long as the number of errors remains low, the inadequate CPTs remain a minority, and the algorithm will work. The experiments illustrate this. The existence of an error will slightly decrease \( p_{\text{cpt}} \), but if it remains high enough inadequate CPTs will still be localized.

B. Deactivating Inadequate Fragments

A CPT considered inadequate can be set to a uniform distribution, which effectively renders the fragment connected to the rest of the network via this CPT inactive. Since a fragment related to the rest of the network via an inadequate CPT does not support accurate classification in the given situation, its deactivation at runtime can improve the overall classification accuracy.

Since the fault localization algorithm can fail, occasionally adequate CPTs could be considered inadequate, which can actually reduce the classification quality. However, by considering the properties of the localization algorithm, we can show that it is more likely to encounter cases, for which the classification quality improves. This is especially the case if several fragments have identical topologies and CPTs, which corresponds to models of conditionally independent processes of the same type running in parallel. Such models are especially relevant for monitoring systems, where we might use several sensors of the same type. As it was shown in [12], each sensor corresponds to a conditionally independent network fragment in a BN given some monitored phenomenon.

The improvement of the classification through deactivation of fragments is illustrated with the help of an experiment. We used a BN with a tree topology, branching factor 5 and 4 levels of nodes. The CPTs at every level were identical, such that \( p_{\text{cpt}} = 0.75 \). This network was used to sample data.


Figure 5. Network structure used for the experiment with the real network.

Line shows the effectiveness if we enforce only one alternative state (i.e. if a fragment is inadequate it will always support the same inaccurate state), which on average decreases the consistency measure. The worst case scenario is equivalent to localization in binary state BNs. We expect real networks to be somewhere in between this worst and best case scenario.

B. Real Network

Next, we tested the algorithm on a real network, namely a subtree of the Munin medical diagnosis network [9] (see Figure 5 for the subtree structure). This tree BN is a significant simplification of the problem domain. It was constructed by first manually setting the network structure and then using the EM algorithm [10] to learn the parameters from a data set sampled from the complete network. Obviously, when we would attempt to classify cases using this simple BN, misclassifications will occur because of implicit errors. The question is whether our algorithm can detect these misclassifications and localize their causes.

We applied the algorithm on the tree BN for a set of sample cases generated by the complete network. Since the state of all (hidden) variables in all cases was known, we knew which CPTs were inadequate. On the tree network, the algorithm found 75.7% of all inadequate CPTs, while producing 20.9% false positives, which confirms that the algorithm can be effective in a real world setting, even for an oversimplified model.

VI. Applications in Bayesian Fusion Systems

The presented approach to fault localization is suitable for BNs with many conditionally independent network fragments given classification variables of interest. Such BNs are relevant especially for monitoring systems which are based on the fusion of large quantities of heterogeneous and uncertain information [11]. Namely, monitoring processes can often be viewed as causal stochastic processes, where hidden events cause observations with certain probability. In such settings, we can assume that each event is significantly influenced only by a small fraction of other events. For example, in monitoring systems, hardware components or reports from one sensor do not influence hardware or reports from another sensor. The corresponding BNs typically consist of several network fragments which are conditionally independent given small sets of variables. In addition, monitoring processes can often be adequately represented by BNs featuring tree topologies with large branching factor, which is especially the case if we use many heterogeneous information sources.
sampled data sets, consisting of 5000 cases, were fed to two classifiers. Both classifiers were based on the BN identical to the generative model. For one classifier we used the fault localization and deactivated inadequate fragments. Compared to the classifier using the unaltered BN, the average posterior probability of the true state of the classification variable was higher (0.81 instead of 0.76). Finally, of the cases that were misclassified by the unaltered BN, 11% got correctly classified after deactivation of inadequate CPTs. In contrast, a correct classification became a misclassification after deactivation in only 2% of cases.

Fragment deactivation is useful for the mitigation of faulty information sources. We assume that a sensor failure is a rare event. Consequently, if a sensor is broken, the fragment corresponding to the sensor is inadequate. If one or a few of the existing sensors are broken, we can localize the corresponding CPTs and mitigate the impact of broken sensors by deactivating the corresponding network fragments.

VII. DISCUSSION

A. Related Work

Several authors have addressed the problems with reliable inference and modeling robustness. Sensitivity based approaches focus on the identification of modeling components that have a significant impact on the inference process [13], [14]. We must take special care of such components, since eventual modeling faults will have a great impact as well. Sensitivity analysis is carried out prior to the operation and deals with the accuracy of the generalizations.

Another class of approaches, including ours, is focusing on determination of the model quality or performance in a given situation at runtime. The central idea of our approach is observation of the consistency of the model’s runtime reinforcements, which is different from common approaches to runtime analysis of BNs, as for example, data conflict [2] and straw models [3], [4]. The data conflict approach is based on the assumption that given an adequate model all observations should be correlated and \( p(e_1, \ldots, e_n) > p(e_1) \cdots p(e_n) \). If this inequality is not satisfied then this is an indication that the model does not ‘fit’ the current set of observations [5]. A generalization of this method, [4], is based on the use of straw models. Simpler (straw) models are constructed through partial marginalization, which, in a coherent situation, should be less probable than the original model. Situations in which the evidence is very unlikely under the original model and more probable under the straw model indicate a data conflict. While these approaches can handle more general BNs than our method, their disadvantage is that the conflict scores are difficult to interpret; it is unclear at which score an action should be undertaken, or what the probability is that a positive score indicates an error. The properties of the method under different model structures are also unknown.

In addition, many of the other approaches focus on the net-performance of models and do not directly support detection of inaccurate parts of a model [15]. An exception is the approach [16] based on logarithmic penalty scoring rules. However, in this case the scores can be determined only for the nodes corresponding to observable events, while we reason about the nodes modeling hidden events.

B. Conclusion

We have presented an approach to error detection for BNs that are used for classification. This was done through the following steps: First, we identified a partitioning of a BN such that each fragment has an independent influence on a classification variable through its corresponding factor. The accuracy of a classification was then related to the adequacy of factors. We introduced a measure to monitor the consistency among multiple network fragments, and showed that we can find thresholds on this measure such that we can deduce the probability that a fragment is (in)adequate. Finally, an algorithm was presented that combines the consistency measures at different nodes in the network in order to determine whether the CPT between the nodes is adequate.

We showed that the results from fault localization can be used in several ways, such as localization of erroneous modeling parameters, faulty information sources and modeling components that do not support accurate classification in a particular case. Furthermore, we established a lower bound to the algorithm’s effectiveness, which we illustrated to converge asymptotically to 1 for network topologies with increasing branching factors.

One might question the assumption about large branching factors. However, there exist applications where this is the case, as for example the Distributed Perception Networks [17], which deals with hundreds or thousands of observations, each corresponding to a fragment in a BN.

In this paper we mainly focussed on tree-like networks. It should be noted that non-trees can be converted to trees by grouping variables into hypernodes, as is for example done during compilation of junction trees [5]. More discussion on the implications of this can be found in [8].

REFERENCES


