You Only Derive Once (YODO): Automatic Differentiation for Efficient Sensitivity Analysis in Bayesian Networks

Rafael Ballester-Ripoll
rafael.ballester@ie.edu

Manuele Leonelli
manuele.leonelli@ie.edu

School of Science and Technology, IE University, Madrid, Spain

Abstract

Sensitivity analysis measures the influence of a Bayesian network’s parameters on a quantity of interest defined by the network, such as the probability of a variable taking a specific value. In particular, the so-called sensitivity value measures the quantity of interest’s partial derivative with respect to the network’s conditional probabilities. However, finding such values in large networks with thousands of parameters can become computationally very expensive. We propose to use automatic differentiation combined with exact inference to obtain all sensitivity values in a single pass. Our method first marginalizes the whole network once using e.g. variable elimination and then backpropagates this operation to obtain the gradient with respect to all input parameters. We demonstrate our routines by ranking all parameters by importance on a Bayesian network modeling humanitarian crises and disasters, and then show the method’s efficiency by scaling it to huge networks with up to 100’000 parameters. An implementation of the methods using the popular machine learning library PyTorch is freely available.

Keywords: Automatic differentiation; Bayesian networks; Sensitivity analysis; Markov random fields; Tensor networks.

1. Introduction

Probabilistic graphical models, and specifically Bayesian networks (BNs), are a class of models that are widely used for risk assessment of complex operational systems in a variety of domains. The main reason for their success is that they provide an efficient as well as intuitive framework to represent the joint probability of a vector of variables of interest using a simple graph. Their use to assess the reliability of engineering, medical and ecological systems, among many others, is becoming increasingly popular. Sensitivity analysis is a critical step for any applied real-world analysis to assess the importance of various risk factors and to evaluate the overall safety of the system under study (see e.g. Goerlandt and Islam 2021, Makaba et al. 2021, Zio et al. 2022, for some recent examples).

As noticed by Rohmer (2020), sensitivity analysis in BNs is usually local, in the sense that it measures the effect of a small number of parameter variations on output probabilities of interest, while other parameters are kept fixed. In the case of a single parameter variation, sensitivity analysis is usually referred to as one-way, otherwise, when more than one parameter is varied, it is called multi-way. Although recently there has been an increasing interest in proposing global sensitivity methods for BNs measuring how different factors jointly influence some function of the model’s output (see e.g. Ballester-Ripoll and Leonelli 2022, Li and Mahadevan 2018), the focus of this paper still lies in one-way sensi-
tivity methods. However, extensions to multi-way local methods are readily available and discussed in Sec. 5.

One-way local sensitivity analysis in BNs can be broken down into two main steps. First, some parameters of the model are varied and the effect of these variations on output probabilities of interest is investigated. For this purpose, a simple mathematical function, usually termed sensitivity function (Castillo et al., 1997; Coupé and van der Gaag, 2002). Furthermore, some specific properties of such a function can be computed, as for instance, the sensitivity value or the vertex proximity, which give an overview of how sensitive the probability of interest is to variations of the associated parameter (van der Gaag et al., 2007). Such measures are reviewed below in Sec. 2. Second, once parameter variations are chosen, the effect of these is summarized by a distance or divergence measure between the baseline and the varied distributions underlying the BN, most commonly the Chan-Darwiche distance (Chan and Darwiche, 2005) or the well-known Kullback-Leibler divergence.

As demonstrated by Kwisthout and van der Gaag (2008), the derivation of both the sensitivity function and its associated properties is computationally very demanding. Here we provide a novel, computationally highly-efficient method to compute many sensitivity measures of interest which takes advantage of backpropagation. The method works thanks to automatic differentiation: by computing the probability of interest once and working out the derivatives of each step along the way in reverse order (the so-called backward pass), one obtains the gradient w.r.t. all model parameters at once (Baydin et al., 2017) (in our case, the network’s conditional probability tables). Once we have the gradient, we bake in the proportional covariation principle (Laskey, 1995) to obtain the desired sensitivity metrics. Our algorithm is demonstrated in a BN modeling humanitarian crises and disasters (Sec. 4), and an extensive simulation study shows its efficiency by processing huge networks in a few seconds. We have open-sourced a Python implementation using the popular machine learning library PyTorch [1], contributing to the recent effort of promoting sensitivity analysis (Douglas-Smith et al., 2020).

2. Bayesian Networks and Sensitivity Analysis

A BN is a probabilistic graphical model defining a factorization of the probability distribution of a random vector by means of a directed acyclic graph (DAG). More formally, let \([n] = \{1, \ldots, n\}\) and \(Y = (Y_i)_{i \in [n]}\) be a random vector of interest with sample space \(\mathcal{Y} = \times_{i \in [n]} \mathcal{Y}_i\). A BN defines the probability distribution \(P(Y = y)\), for \(y \in \mathcal{Y}\), as a product of simpler conditional probability distributions as follows:

\[
P(Y = y) = \prod_{i \in [n]} P(Y_i = y_i \mid Y_{\Pi_i} = y_{\Pi_i}),
\]

where \(Y_{\Pi_i}\) are the parents of \(Y_i\) in the DAG associated to the BN.

The definition of the probability distribution over \(Y\), which would require defining \(#\mathcal{Y} - 1\) probabilities, is thus simplified in terms of one-dimensional conditional probability distribution. The coefficients of these functions are henceforth referred to as the parameters

---

1. Available at https://github.com/rballester/gmtorch
θ of the model. The DAG structure may be either elicited from experts or learned from data using structural learning algorithms, and the associated parameters θ can be either elicited from experts as well or learned using frequentist or Bayesian approaches. No matter the method used, we assume that a value for these parameters θ has been determined which we refer to as the baseline value and denoted as θ₀.

In practical applications it is fundamental to extensively assess the implications of the chosen parameter values θ₀ to outputs of the model. In the context of BNs this study is usually referred to as sensitivity analysis, which can actually be further used during the model building process as showcased by Coupé et al. (2000). Let Y_O be an output variable of interest and Y_E be evidential variables, those that may be observed. The interest is in then studying how P(Y_O = y_O | Y_E = y_E) varies when a parameter θ_i is varied. In particular, P(Y_O = y_O | Y_E = y_E) seen as a function of θ_i is called sensitivity function and denoted as f_O,E(θ_i).

2.1 Proportional Covariation

Notice that when an input θ_i is varied from its baseline value θ₀_i then the parameters from the same conditional probability distribution need to covary to respect the sum-to-one condition of probabilities. When variables are binary, this is automatic since one parameter must be equal to one minus the other, but for variables taking more than two levels this co-variation can be done in several ways (Renooij, 2014). We henceforth assume that whenever a parameter is varied from its baseline value θ₀_i to a new value θ_i, then every parameter θ_j from the same conditional probability distribution is proportionally covaried (Laskey, 1995):

θ_j(θ_i) = \frac{1 - θ_i}{1 - θ₀_i} \theta₀_j.

(1)

Proportional covariation has been studied extensively and its choice is motivated by a wide array of theoretical properties (Chan and Darwiche, 2005; Leonelli et al., 2017; Leonelli and Riccomagno, 2018).

Under the assumption of proportional covariation, Castillo et al. (1997) and Coupé and van der Gaag (2002) demonstrated that the sensitivity function is the ratio of two linear functions:

f_O,E(θ_i) = \frac{c_1 θ_i + c_2}{c_3 θ_i + c_4},

(2)

where c₁, c₂, c₃, c₄ ∈ \mathbb{R}_+. van der Gaag et al. (2007) noticed that the above expression actually coincides with the fragment of a rectangular hyperbola, which can be generally written as f_O,E(θ_i) = \frac{r}{s - t} + t where s = -\frac{c_3}{c_4}, t = \frac{c_2}{c_3} and r = \frac{c_2}{c_3} + st.

2.2 Sensitivity Value

The sensitivity value describes the effect of infinitesimally small shifts in the parameter’s baseline value on the probability of interest and is defined as the absolute value of the first derivative of the sensitivity function at the baseline value of the parameter, i.e. |f'(θ₀_i)|.
This can be found by simply differentiating the sensitivity function as

$$|f'_{O,E}(\theta_i^0)| = \frac{|c_1 c_4 - c_2 c_3|}{(c_3 \theta_i^0 + c_4)^2}. \quad (3)$$

The higher the sensitivity value, the more sensitive the output probability to small changes in the baseline value of the parameter. As a rule of thumb, parameters having a sensitivity value larger than one may require further investigation [van der Gaag et al., 2007].

Notice that when $Y_E$ is empty, i.e. the output probability of interest is a marginal probability, then the sensitivity function is linear in $\theta_i$ and the sensitivity value is the same no matter what the baseline $\theta_i^0$ was. Therefore in this case the absolute value of the gradient is sufficient to quantify the effect of a parameter to an output probability of interest.

2.3 Vertex Proximity

van der Gaag et al. [2007] further noticed that parameters for which the sensitivity value is small may still be such that the conditional output probability of interest is very sensitive to their variations. This happens when the baseline parameter value is close to the vertex of the sensitivity function, defined as the point $\theta_i^v$ at which the sensitivity value is equal to one, i.e. $|f'_{O,E}(\theta_i^v)| = 1$. The vertex can be derived from the equation of the sensitivity function as

$$\theta_i^v = \begin{cases} s + \sqrt{|r|}, & \text{if } s < 0, \\ s - \sqrt{|r|}, & \text{if } s > 0. \end{cases}$$

Notice that the case $s = 0$ is not contemplated since it would coincide to a linear sensitivity function, not an hyperbolic one.

Vertex proximity is defined as the absolute difference $|\theta_i^0 - \theta_i^v|$. The smaller the vertex proximity, the more sensitive the output probabilities may be to variations of the parameter, even when the sensitivity value is small.

2.4 Other Metrics

Given the coefficients $c_1, \ldots, c_4$ of Eq. (2), it is straightforward to derive any property of the sensitivity function besides the sensitivity value and the vertex proximity. Here we propose the use of two additional metrics. The first is the absolute value of the second derivative of the sensitivity function at the baseline parameter value, which can be easily computed as:

$$|f''_{O,E}(\theta_i^0)| = \frac{2c_3 |c_1 c_4 - c_2 c_3|}{(c_3 \theta_i^0 + c_4)^3}.$$  

Similarly to the sensitivity value, high values of the second derivative at $\theta_i^0$ indicate parameters that could highly impact the probability of interest.

The second measure is the maximum of the first derivative of the sensitivity function over the interval $[0, 1]$ in absolute value, which we find easily by noting that the denominator of Eq. (3) is a parabola:

$$\max_{\theta_i \in [0, 1]} |f'_{O,E}(\theta_i)| = \begin{cases} \infty & \text{if } -c_4/c_3 \in [0, 1] \\ \max\{|c_1 c_4 - c_2 c_3|/c_4^2, |c_1 c_4 - c_2 c_3|/(c_3 + c_4)^2\} & \text{otherwise.} \end{cases}$$
Again high values indicate parameters whose variations can lead to a significant change in the output probability of interest.

3. The YODO Method

3.1 First Case: Marginal Probability as a Function of Interest

Suppose \( f(\theta_i) = P(Y_O = y_O) = c_1 \theta_i + c_2 \) assuming proportional covariation as \( \theta_i \) moves. Let \( \theta_{j_1}, \ldots, \theta_{j_n} \) be the other parameters of the same conditional PMF as \( \theta_i \), i.e. they are all bound by the sum-to-one constraint \( \theta_i + \theta_{j_1} + \cdots + \theta_{j_n} = 1 \). First, we rewrite \( f_{O,E} \) as

\[
 f_{O,E}(\theta_i) = g(\theta_i, \theta_{j_1}(\theta_i), \ldots, \theta_{j_n}(\theta_i))
\]

and we will show how to obtain \( f'_{O,E}(\theta_i) \) provided that we can compute the gradient \( \nabla g \) with respect to symbols \( \theta_i, \theta_{j_1}, \ldots, \theta_{j_n} \) (Sec. 3.3 for details on the latter).

By the generalized chain rule, it holds that

\[
 f'_{O,E}(\theta_i) = \frac{\partial g}{\partial \theta_i} \cdot 1 + \frac{\partial g}{\partial \theta_{j_1}} \cdot \frac{d\theta_{j_1}}{d\theta_i} + \cdots + \frac{\partial g}{\partial \theta_{j_n}} \cdot \frac{d\theta_{j_n}}{d\theta_i}.
\]

By deriving Eq. (1), we have that for all \( 1 \leq m \leq n \):

\[
 \frac{d\theta_{j_m}}{d\theta_i} = -\frac{\theta_{j_m}^0}{1 - \theta_i^0}
\]

and, therefore,

\[
 f'_{O,E}(\theta_i) = \frac{\partial g}{\partial \theta_i} - \frac{(\partial g/\partial \theta_{j_1}) \cdot \theta_{j_1}^0 + \cdots + (\partial g/\partial \theta_{j_n}) \cdot \theta_{j_n}^0}{1 - \theta_i^0}.
\]

Last, since \( f_{O,E}(\theta_i) = P(Y_O = y_O) = c_1 \theta_i + c_2 \), we easily find the parameters \( c_1, c_2 \):

\[
 \begin{cases}
 c_1 = f'_{O,E}(\theta_i) \\
 c_2 = P(Y_O = y_O) - c_1 \theta_i^0
\end{cases}
\]

3.2 Second Case: Conditional Probability as a Function of Interest

When \( f_{O,E}(\theta_i) = P(Y_O = y_O \mid Y_E = y_E) = P(Y_O = y_O, Y_E = y_E) / P(Y_E = y_E) \), we simply repeat the procedure from Sec. 3.1 twice:

1. We first apply it to \( P(Y_O = y_O, Y_E = y_E) \) to obtain \( c_1 \) and \( c_2 \);

2. we then apply it to \( P(Y_E = y_E) \) to obtain \( c_3 \) and \( c_4 \).

3.3 Computing the Gradient \( \nabla g \)

Let \( Y_K = y_K \) be a subset of the network variables taking some evidence values (this could be \( K = O \) or \( K = O \cup E \), hence we cover the two cases above).

We start by moralizing the BN into a Markov random field (MRF) \( M \). This marries the parents of each variable together and, for each conditional probability table (now called
potential), drops the sum-to-one constraint; see e.g. [Darwiche 2009] for more details. Next, we impose the evidence $Y_K = y_K$ by defining $M_{Y_K = y_K}$ as a new MRF that results from substituting each potential $\Phi_{i_1, \ldots, i_M}(x_{i_1}, \ldots, x_{i_M})$ by a new potential $\hat{\Phi}_{i_1, \ldots, i_M}$ defined as follows:

$$
\hat{\Phi}_{i_1, \ldots, i_M}(Y_{i_1} = x_{i_1}, \ldots, Y_{i_M} = x_{i_M}) =
\begin{cases}
0 & \text{if } \exists m, k \mid i_m = k \land x_{i_m} \neq y_{i_m} \\
\Phi_{i_1, \ldots, i_M}(Y_{i_1} = x_{i_1}, \ldots, Y_{i_M} = x_{i_M}) & \text{otherwise}
\end{cases}
$$

In other words, we copy the original potential but zero-out all entries that are incompatible with the assignment of values $Y_K = y_K$. See Tab. 1 for an example using a bivariate potential.

<table>
<thead>
<tr>
<th>$Y_2 = 1$</th>
<th>$Y_2 = 2$</th>
<th>$Y_2 = 3$</th>
<th>$Y_2 = 1$</th>
<th>$Y_2 = 2$</th>
<th>$Y_2 = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_1 = 1$</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>$Y_1 = 1$</td>
<td>0</td>
</tr>
<tr>
<td>$Y_1 = 2$</td>
<td>0.3</td>
<td>0.5</td>
<td>0.2</td>
<td>$Y_1 = 2$</td>
<td>0</td>
</tr>
<tr>
<td>$Y_1 = 3$</td>
<td>0.1</td>
<td>0.2</td>
<td>0.7</td>
<td>$Y_1 = 3$</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) $\Phi_{1,2}(y_1,y_2)$

<table>
<thead>
<tr>
<th>$Y_2 = 1$</th>
<th>$Y_2 = 2$</th>
<th>$Y_2 = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_1 = 1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$Y_1 = 2$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$Y_1 = 3$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) $\hat{\Phi}_{1,2}(y_1,y_2)$

Table 1: Left: example potential of an MRF $M$ for variables $Y_1$ and $Y_2$, each with three levels \{1, 2, 3\}. Right: corresponding potential for $M_{Y_2 = 3}$.

Intuitively, the modified MRF $M_{Y_K = y_K}$ represents the unnormalized probability for all variable assignments that are compatible with $Y_K = y_K$. In particular, if $M_{Y_K}$ denotes the marginalization of a network $M$ over all variables in $Y_K$, we have that $(M_{Y_K = y_K})_{Y} = P(Y_K = y_K)$. In other words, computing $g$ reduces to marginalizing our MRF. In this paper we marginalize it exactly using the variable elimination (VE) algorithm; see e.g. [Darwiche 2009]. This method is clearly differentiable w.r.t. all parameters $\theta$ since VE only relies on variable summation and factor multiplication. Any other differentiable inference algorithm could be used as well. This step, evaluating the function $g$, is known as the forward pass in the neural network literature. Next, we backpropagate the previous operation (a step also known as the backward pass) to build the gradient $\nabla g$. Crucially, note that backpropagation yields $\partial g / \partial \theta$ for every parameter $\theta \in \theta$ of the network at once, not just an individual $\theta_i$. Last, we obtain parameters $c_1, \ldots, c_4$ as detailed before, and use them to compute the metrics of Secs. 2.2, 2.3, and 2.4 for each $\theta_i$.

Note the advantages of this approach as compared to other alternatives. For example, symbolically deriving the gradient of $g$ would be cumbersome and would depend on the target network topology and definition of the probability of interest [Darwiche 2003]. Automatic differentiation avoids this by evaluating the gradient numerically using the chain rule. Furthermore, finding the gradient using finite differences would require evaluating $g$ twice per parameter $\theta_i$. In contrast, automatic differentiation only requires a forward and backward pass to find the entire gradient: in our experiments, this took roughly the time of just two marginalization operations (see next section).
4. Results

We overview first the insights revealed by our method when applied on a 21-node Bayesian network of interest; we then study the method’s scalability by testing it on large networks with hundreds of nodes and arcs and up to $10^5$ parameters.

4.1 Software and Hardware Used

In order to perform variable elimination efficiently, we note that the problem of graphical model marginalization is equivalent to that of tensor network contraction (Robeva and Seigal, 2018), and use the library opt_einsum (Smith and Gray, 2018) which offers optimized heuristics for the latter. As backend we use the state-of-the-art machine learning library PyTorch (Paszke et al., 2019), version 1.11.0, to do all operations between tensors and then perform backpropagation on them. We use pgmpy (Ankan and Panda, 2015) for reading and moralizing BNs. All experiments were run on a 4-core i5-6600 3.3GHz Intel workstation with 16GB RAM.

4.2 Risk Assessment for Humanitarian Crises and Disasters

Similarly to Qazi and Simsekler (2021), we construct a BN model to assess the country-level risk associated with humanitarian crises and disasters. The data was collected from INFORM (INFORM, 2022) and consists of 20 drivers of disaster risk covering natural, human, socio-economic, institutional and infrastructure factors that influence the country-level risk of a disaster, together with a final country risk index which summarizes how exposed a country is to the possibility of a humanitarian disaster. A full list of the variables can be found at INFORM (2022). All variables take values between zero and ten and have been discretized into three categories (low/0, medium/1, high/2) using the equal-length method. The dataset comprises 190 countries.

A BN is learned using the hc function of the bnlearn package and is reported in Fig. 1. As an illustration of the YODO method, we compute here all sensitivity measures for the conditional probability of a high risk of disaster (RISK = 2) conditional on a high risk of flooding (FLOOD = 2). Computing all metrics for all 183 network parameters with our method took only 0.055 seconds. The results are reported in Tab. 2 for the 20 most influential parameters according to the sensitivity value. It can be noticed that the most influential parameters come from the conditional distributions of the overall risk given the development and deprivation index (D_AND_D), as well as from the conditional distribution of the flooding index given a projected conflict risk index (PCR) equal to low. As an additional illustration, Fig. 2 reports the sensitivity value of the parameters for the output conditional probability of an overall high risk given a high risk of earthquake. The blue color is associated to positive values of the sensitivity value, the red color for negative ones. Out of 183 network parameters, 30 had a sensitivity value of zero, meaning that they had no effect on the probability of interest.

4.3 Performance Study over Medium to Very Large Networks

We further run our method over the 10 Bayesian networks considered in Scutari et al. (2019). As a baseline we use numerical estimation of each sensitivity value via finite differ-
Figure 1: BN learned over the INFORM [2022] dataset for country-level disaster risk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Sensitivity value</th>
<th>Proximity 2nd deriv</th>
<th>Largest 1st deriv</th>
</tr>
</thead>
<tbody>
<tr>
<td>RISK = high</td>
<td>D_AND_D = low</td>
<td>0.0012 0.9140</td>
<td>0.0566 1.4378</td>
<td>0.9161</td>
</tr>
<tr>
<td>FLOOD = high</td>
<td>PCR = low</td>
<td>0.1070 0.7220</td>
<td>0.0334 4.0591</td>
<td>1.4750</td>
</tr>
<tr>
<td>FLOOD = medium</td>
<td>PCR = low</td>
<td>0.4690 0.6450</td>
<td>0.7185 3.2382</td>
<td>0.0000</td>
</tr>
<tr>
<td>FLOOD = low</td>
<td>PCR = low</td>
<td>0.4250 0.6450</td>
<td>0.7185 3.2382</td>
<td>0.8714</td>
</tr>
<tr>
<td>RISK = high</td>
<td>D_AND_D = high</td>
<td>0.3400 0.5550</td>
<td>0.7310 3.8787</td>
<td>0.7140</td>
</tr>
<tr>
<td>RISK = high</td>
<td>D_AND_D = medium</td>
<td>0.0686 0.4670</td>
<td>1.0020 0.2950</td>
<td>0.4880</td>
</tr>
<tr>
<td>EPIDEMIC = high</td>
<td>HEALTH_COND = low</td>
<td>0.1480 0.2950</td>
<td>1.1670 0.2310</td>
<td>0.3320</td>
</tr>
<tr>
<td>D_AND_D = high</td>
<td>EPIDEMIC = medium</td>
<td>0.0742 0.2380</td>
<td>1.8344 0.1330</td>
<td>0.2490</td>
</tr>
<tr>
<td>PCR = high</td>
<td>RISK = medium</td>
<td>0.2780 0.2260</td>
<td>0.60 0.3950</td>
<td>0.3940</td>
</tr>
<tr>
<td>PCR = high</td>
<td>RISK = low</td>
<td>0.0266 0.2040</td>
<td>0.6940 3.2230</td>
<td>0.2310</td>
</tr>
<tr>
<td>FLOOD = high</td>
<td>PCR = high</td>
<td>0.5090 0.1960</td>
<td>0.4750 0.4610</td>
<td>1.2110</td>
</tr>
<tr>
<td>FLOOD = high</td>
<td>PCR = medium</td>
<td>0.1360 0.1670</td>
<td>0.7870 0.2500</td>
<td>0.2060</td>
</tr>
<tr>
<td>D_AND_D = high</td>
<td>EPIDEMIC = high</td>
<td>0.7870 0.1590</td>
<td>4.1590 0.0459</td>
<td>0.2020</td>
</tr>
<tr>
<td>D_AND_D = high</td>
<td>EPIDEMIC = low</td>
<td>0.0411 0.1530</td>
<td>2.9840 0.0625</td>
<td>0.1560</td>
</tr>
<tr>
<td>RISK = low</td>
<td>D_AND_D = high</td>
<td>0.0028 0.1510</td>
<td>3.2190 0.0796</td>
<td>0.2740</td>
</tr>
<tr>
<td>RISK = medium</td>
<td>OTHER_VULN_GROUPS = low</td>
<td>0.0500 0.1510</td>
<td>3.0260 0.0601</td>
<td>0.1540</td>
</tr>
<tr>
<td>HEALTH_COND = low</td>
<td>OTHER_VULN_GROUPS = low</td>
<td>0.9949 0.15</td>
<td>3.0360 0.0606</td>
<td>0.1540</td>
</tr>
<tr>
<td>PCR = low</td>
<td>RISK = high</td>
<td>0.00521 0.15</td>
<td>3.0230 0.0609</td>
<td>0.2360</td>
</tr>
<tr>
<td>D_AND_D = medium</td>
<td>EPIDEMIC = high</td>
<td>0.1760 0.1480</td>
<td>5.3930 0.0334</td>
<td>0.1800</td>
</tr>
<tr>
<td>PCR = high</td>
<td>RISK = high</td>
<td>0.9430 0.1480</td>
<td>3.0920 0.0588</td>
<td>0.2240</td>
</tr>
<tr>
<td>PCR = medium</td>
<td>RISK = high</td>
<td>0.0521 0.1460</td>
<td>3.1440 0.0573</td>
<td>0.2200</td>
</tr>
<tr>
<td>PCR = medium</td>
<td>RISK = low</td>
<td>0.0368 0.1440</td>
<td>2.8590 0.0626</td>
<td>0.2310</td>
</tr>
<tr>
<td>FLOOD = low</td>
<td>PCR = medium</td>
<td>0.1030 0.1440</td>
<td>0.9550 0.1870</td>
<td>0.1650</td>
</tr>
<tr>
<td>FLOOD = medium</td>
<td>PCR = medium</td>
<td>0.7620 0.1440</td>
<td>0.9550 0.1870</td>
<td>0.5650</td>
</tr>
<tr>
<td>RISK = medium</td>
<td>D_AND_D = high</td>
<td>0.6400 0.1370</td>
<td>1.9900 0.0860</td>
<td>0.1750</td>
</tr>
<tr>
<td>PCR = low</td>
<td>RISK = medium</td>
<td>0.4220 0.12</td>
<td>1.3990 0.1120</td>
<td>0.2260</td>
</tr>
<tr>
<td>HEALTH_COND = high</td>
<td>OTHER_VULN_GROUPS = low</td>
<td>0.009083 0.1130</td>
<td>4.3000 0.0348</td>
<td>0.1130</td>
</tr>
<tr>
<td>OTHER_VULN_GROUPS = low</td>
<td>RECENT_SHOCKS = low</td>
<td>0.8200 0.1120</td>
<td>4.4180 0.0337</td>
<td>0.1180</td>
</tr>
<tr>
<td>OTHER_VULN_GROUPS = low</td>
<td>RECENT_SHOCKS = low</td>
<td>0.1730 0.1110</td>
<td>4.4540 0.0333</td>
<td>0.1170</td>
</tr>
<tr>
<td>PCR = low</td>
<td>RISK = low</td>
<td>0.9470 0.1090</td>
<td>1.5780 0.0928</td>
<td>0.1150</td>
</tr>
</tbody>
</table>

Table 2: Four sensitivity metrics for the top 30 parameters of the humanitarian crisis network, when the probability of interest is $P(RISK = high | FLOOD = high)$.

ences, whereby we slightly perturb each parameter $\theta_i$ and measure the impact on $f$. As a probability of interest we set $P(A = a | B = b)$, where $A, B, a, b$ were two variables and two levels picked at random, respectively, and each timing is the average of three independent runs. Results are reported in Tab. 3 which shows that YODO outperforms the baseline by several orders of magnitude.
Figure 2: Top 20 most influential parameters for the humanitarian crisis network, color-coded by the sign of $f'(\theta_i)$. The probability of interest is $P(\text{RISK} = \text{high} \ | \ \text{EARTHQUAKE} = \text{high})$.

Table 3: Our method was applied to 10 Bayesian networks, here sorted by number of nodes. All times are in seconds. The times for the baseline (second-to-last column) were estimated as the total number of parameters in the network times the time needed to numerically estimate one sensitivity value. Treewidths were found with the NetworkX graph library [Hagberg et al., 2008].

5. Discussion

We demonstrated the use of automatic differentiation in the area of BNs and more specifically in the study of how sensitive they are to variations of their parameters. The novel
algorithms are freely available in Python and are planned to be included in the next release of the \texttt{bnmonitor} R package (Leonelli et al., 2021). Their efficiency was demonstrated through a simulation study and their use in practice was illustrated through a BN in the field of risk assessment for humanitarian crises.

Although YODO is specifically designed to compute the coefficients of the sensitivity function in Eq. (2), it further addresses two additional problems in sensitivity analysis. First, it is able to quickly find which parameters do have an effect on the output probability of interest, which is usually called the parameter sensitivity set (Coupé and van der Gaag, 2002). Second, we identify whether a parameter change leads to a monotonically increasing or decreasing sensitivity function, as already addressed in Bolt and Renooij (2017). Although the above-cited works only require the structure of the network, YODO yields an efficient way to tackle the same problems.

Future Work

Because of the space constraint we only focused on one-way sensitivity analysis but, because of their efficiency, the proposed methods could be generalized to multi-way sensitivity analysis where more than one parameter is varied simultaneously. Bolt and Renooij (2014) introduced the maximum/minimum n-way sensitivity value which bounds the effect of n-way variations of parameters and demonstrated that it can be easily derived from the sensitivity values of one-way sensitivity analyses. Therefore, our methods could be extended to also efficiently compute the joint effect of variations of parameters, known to be computationally challenging (Chan and Darwiche, 2004; Kjaerulf and van der Gaag, 2000). Another possible extension of the algorithms introduced here would be to compute the so-called admissible deviation (van der Gaag and Renooij, 2001). This consists of finding a pair of numbers \((\alpha, \beta)\) that describe the shifts to smaller values and to larger values, respectively, that are allowed in the parameter under study without inducing a change in the most likely value of the output variable. For a parameter with a baseline value of \(\theta_0\), the admissible deviation \((\alpha, \beta)\) thus indicates that the parameter can be safely varied within the interval \((\theta_0 - \alpha, \theta_0 + \beta)\). These values can be straightforwardly found by identifying the intersections of the sensitivity functions associated to different values of the output variable.

References


