A Symbolic Approach to Explaining Bayesian Network Classifiers

Andy Shih and **Arthur Choi** and **Adnan Darwiche**
Computer Science Department, University of California, Los Angeles
{andyshih,aychoi,darwiche}@cs.ucla.edu

Abstract

We propose an approach for explaining Bayesian network classifiers, which is based on compiling such classifiers into decision functions that have a tractable and symbolic form. We introduce two types of explanations for why a classifier may have classified an instance positively or negatively and suggest algorithms for computing these explanations. The first type of explanation identifies a minimal set of the currently active features that is responsible for the current classification, while the second type of explanation identifies a minimal set of features whose current state (active or not) is sufficient for the classification. We consider in particular the compilation of Naive and Latent-Tree Bayesian network classifiers into Ordered Decision Diagrams (ODDs), providing a context for evaluating our proposal using case studies and experiments based on classifiers from the literature.

1 Introduction

Recent progress in artificial intelligence and the increased deployment of AI systems have led to highlighting the need for *explaining* the decisions made by such systems, particularly classifiers; see, e.g., [Ribeiro *et al.*, 2016b; Elenberg *et al.*, 2017; Lundberg and Lee, 2017; Ribeiro *et al.*, 2018].[1] For example, one may want to explain *why* a classifier decided to turn down a loan application, or rejected an applicant for an academic program, or recommended surgery for a patient. Answering such *why?* questions is particularly central to assigning blame and responsibility, which lies at the heart of legal systems and may be required in certain contexts.[2]

In this paper, we propose a *symbolic* approach to explaining Bayesian network classifiers, which is based on the following observation. Consider a classifier that labels a given instance either positively or negatively based on a number of discrete features. Regardless of how this classifier is implemented, e.g., using a Bayesian network, it does specify a symbolic function that maps features into a yes/no decision (yes for a positive instance). We refer to this function as the classifier's *decision function* since it unambiguously describes the classifier's behavior, independently of how the classifier is implemented. Our goal is then to obtain a symbolic and tractable representation of this decision function, to enable symbolic and efficient reasoning about its behavior, including the generation of explanations for its decisions. In fact, [Chan and Darwiche, 2003] showed how to compile the decision functions of naive Bayes classifiers into a specific symbolic and tractable representation, known as Ordered Decision Diagrams (ODDs). This representation extends Ordered Binary Decision Diagrams (OBDDs) to use multi-valued variables (discrete features), while maintaining the tractability and properties of OBDD [Bryant, 1986; Meinel and Theobald, 1998; Wegener, 2000].

We show in this paper how compiling decision functions into ODDs can facilitate the efficient explanation of classifiers and propose two types of explanations for this purpose.

The first class of explanations we consider are *minimum-cardinality explanations*. To motivate these explanations, consider a classifier that has diagnosed a patient with some disease based on some observed test results, some of which were positive and others negative. Some of the positive test results may not be necessary for the classifier's decision: the decision would remain intact if these test results were negative. A minimum-cardinality explanation then tells us which of the positive test results are the culprits for the classifier's decision, i.e., a minimal subset of the positive test results that is sufficient for the current decision.

The second class of explanations we consider are *prime-implicant explanations*. These explanations answer the following question: what is the smallest subset of features that renders the remaining features irrelevant to the current decision? In other words, which subset of features—when fixed—would allow us to arbitrarily toggle the values of other features, while maintaining the classifier's decision?

This paper is structured as follows. In Section 2, we review the compilation of naive Bayes classifiers into ODDs, and propose a new algorithm for compiling latent-tree classifiers into ODDs. In Section 3, we introduce minimum-cardinality explanations, propose an algorithm for computing them, and

---

[1]It is now recognized that opacity, or lack of explainability is "one of the biggest obstacles to widespread adoption of artificial intelligence" (The Wall Street Journal, August 10, 2017).

[2]See, for example, the EU general data protection regulation, which has a provision relating to explainability, https://www.privacy-regulation.eu/en/r71.htm.
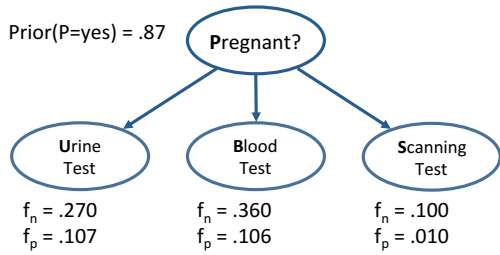
Figure 1: A naive Bayes classifier, specified using the class prior, in addition to the false positive (f_p) and false negative (f_n) rates of features. The class variable and features are all binary.
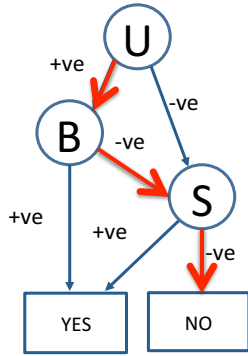

Figure 2: An OBDD (decision function) of the classifier in Figure 1.

provide a case study on a real-world classifier. In Section 4, we do the same for prime-implicant explanations. In Section 5, we discuss the relationship between the two types of explanations and show that they coincide for monotone classifiers. We then follow by a discussion of related work in Section 6 and finally close in Section 7.

2 Compiling Bayesian Network Classifiers

Consider Figure 1 which depicts a naive Bayes classifier for detecting pregnancy. Given results for the three tests, if the probability of pregnancy passes a given threshold (say 90%), we would then obtain a "yes" decision on pregnancy.

Figure 2 depicts the decision function of this classifier, in the form of an Ordered Binary Decision Diagram (OBDD). Given some test results, we make a corresponding decision on pregnancy by simply navigating the OBDD. We start at the root, which is labeled with the Urine (U) test. Depending on the outcome of this test, we follow the edge labeled positive, or the edge labeled negative. We repeat for the test labeled at the next node. Eventually, we reach a leaf node labeled "yes" or "no," which provides the resulting classification.

The decisions rendered by this OBDD are guaranteed to match those obtained from the naive Bayes classifier. We have thus converted a probabilistic classifier into an equivalent classifier that is symbolic and tractable. We will later see how this facilitates the efficient generation of explanations.

We will later discuss compiling Bayesian network classifiers into ODDs, after formally treating classifiers and ODDs.

2.1 Bayesian Network Classifiers

A *Bayesian network classifier* is a Bayesian network containing a special set of variables: a single *class* variable C and n *feature* variables $\mathbf{X} = \{X_1, \ldots, X_n\}$. The class C is usually a root in the network and the features $\mathbf{X}$ are usually leaves. In this paper, we assume that the class variable is binary, with two values c and $\bar{c}$ that correspond to positive and negative classes, respectively (i.e., "yes" and "no" decisions). An instantiation of variables $\mathbf{X}$ is denoted $\mathbf{x}$ and called an *instance.* A Bayesian network classifier specifying probability distribution $Pr(.)$ will classify an instance $\mathbf{x}$ positively iff $Pr(c \mid \mathbf{x}) \geq T$, where T is called the classification *threshold.*

Definition 1 (Decision Function) *Suppose that we have a Bayesian network classifier with features $\mathbf{X}$, class variable C and a threshold T. Let $f(\mathbf{X})$ be a function that maps instances $\mathbf{x}$ into $\{0, 1\}$. We say that $f(\mathbf{X})$ is the classifier's decision function iff*

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } Pr(c \mid \mathbf{x}) \geq T \\ 0 & \text{otherwise.} \end{cases}$$

Instance $\mathbf{x}$ is positive if $f(\mathbf{x}) = 1$ and negative if $f(\mathbf{x}) = 0$.

The *naive Bayes classifier* is a special type of a Bayesian network classifier, where edges extend from the class to features (no other nodes or edges). Figure 1 depicted a naive Bayes classifier. A *latent-tree classifier* is a tree-structured Bayesian network, whose root is the class variable and whose leaves are the features.

2.2 Monotone Classifiers

The class of *monotone* classifiers is relevant to our discussion, particularly when relating the two types of explanations we shall propose. We will define these classifiers next, while assuming binary features to simplify the treatment. Intuitively, a monotone classifier satisfies the following. A positive instance remains positive if we flip some of its features from 0 to 1. Moreover, a negative instance remains negative if we flip some of its features from 1 to 0.

More formally, consider two instances $\mathbf{x}^\star$ and $\mathbf{x}$. We write $\mathbf{x}^\star \subseteq^1 \mathbf{x}$ to mean: the features set to 1 in $\mathbf{x}^\star$ is a subset of those set to 1 in $\mathbf{x}$. Monotone classifiers are then characterized by the following property of their decision functions, which is well-known in the literature on Boolean functions.

Definition 2 *A decision function $f(\mathbf{X})$ is monotone iff*

$$\mathbf{x}^\star \subseteq^1 \mathbf{x} \quad \text{only if} \quad f(\mathbf{x}^\star) \leq f(\mathbf{x}).$$

One way to read the above formal definition is as follows. If the positive features in instance $\mathbf{x}$ contain those in instance $\mathbf{x}^\star$, then instance $\mathbf{x}$ must be positive if instance $\mathbf{x}^\star$ is positive.

It is generally difficult to decide whether a Bayesian network classifier is monotone; see, e.g., [van der Gaag *et al.*, 2004]. However, if the decision function of the classifier is an OBDD, then monotonicity can be decided in time quadratic in the OBDD size [Horiyama and Ibaraki, 2002].

2.3 Ordered Decision Diagrams

An Ordered Binary Decision Diagram (OBDD) is based on an ordered set of binary variables $\mathbf{X} = X_1, \ldots, X_n$. It is a rooted, directed acyclic graph, with two sinks called the 1-sink and 0-sink. Every node (except the sinks) in the OBDD is labeled with a variable X_i with two outgoing edges, one labeled 1 and the other labeled 0. If there is an edge from a node labeled X_i to a node labeled X_j, then $i < j$. An OBDD is defined over binary variables, but can be extended to discrete variables with arbitrary values. This is called an ODD: a node labeled with variable X_i has one outgoing edge for each value of variable X_i. Hence, an OBDD/ODD can be viewed as representing a function $f(\mathbf{X})$ that maps instances $\mathbf{x}$ into $\{0, 1\}$. Figure 2 depicted an OBDD. Note: in this paper, we use positive/yes/1 and negative/no/0 interchangeably.

An OBDD is a *tractable* representation of a function $f(\mathbf{X})$ as it can be used to efficiently answer many queries about the function. For example, one can in linear time count the number of positive instances $\mathbf{x}$ (i.e., $f(\mathbf{x}) = 1$), called the *models* of f. One can also conjoin, disjoin and complement OBDDs efficiently. This tractability, which carries over to ODDs, will be critical for efficiently generating explanations. For more on OBDDs, see [Meinel and Theobald, 1998; Wegener, 2000].

2.4 Compiling Decision Functions

[Chan and Darwiche, 2003] proposed an algorithm for compiling a naive Bayes classifier into an ODD, while guaranteeing an upper bound on the time of compilation and the size of the resulting ODD. In particular, for a classifier with n features, the compiled ODD has a number of nodes that is bounded by $O(b^{\frac{n}{2}})$ and can be obtained in time $O(nb^{\frac{n}{2}})$. Here, b is the maximum number of values that a variable may have. The actual time and space complexity can be much less, depending on the classifier's parameters and variable order used for the ODD (as observed experimentally).

The algorithm is based on the following insights. Let $\mathbf{X}$ be all features. Observing features $\mathbf{Y} \subset \mathbf{X}$ leads to another naive Bayes classifier, with features $\mathbf{X} \setminus \mathbf{Y}$ and an adjusted class prior. Consider now a decision tree over features $\mathbf{X}$ and a node in the tree that was reached by a partial instantiation $\mathbf{y}$. We annotate this node with the corresponding naive Bayes classifier $N_{\mathbf{y}}$ found by observing $\mathbf{y}$, and then merge nodes with equivalent classifiers—those having equivalent decision functions—as described by [Chan and Darwiche, 2003]. Implementing this idea carefully leads to an ordered decision diagram (ODD) with the corresponding bounds.[3]

Algorithm 1 is a simpler variation on the algorithm of [Chan and Darwiche, 2003]; it has the same complexity bounds, but may be less efficient in practice. It uses procedure `expand-then-merge(., D, X)`, which expands the partial decision graph D by a feature X, then merges nodes that correspond to equivalent classifiers.

Using this procedure, we propose Algorithm 2 for compiling a latent-tree classifier into an ODD. Here's the key insight. Let R be a node in a latent-tree classifier where all

---

[3][Chan and Darwiche, 2003] uses a sophisticated, but conceptually simple, technique for identifying equivalent classifiers.

Algorithm 1 `compile-naive-bayes(N)`

input: A naive Bayes classifier N

output: An ODD for the decision function of N

main:

1: $D \leftarrow$ empty decision graph
2: **for** each feature X of classifier N **do**
3:     $D \leftarrow$ `expand-then-merge`(N, D, X)
4: **return** ODD D

Algorithm 2 `compile-latent-tree(N)`

input: A latent-tree classifier N

output: An ODD for the decision function of N

main:

1: $D \leftarrow$ empty decision graph
2: $R \leftarrow$ root of tree N
3: **while** R has unprocessed children **do**
4:     **if** R has a single internal and unprocessed child C **then**
5:         $R \leftarrow C$
6:     **else**
7:         $C \leftarrow$ child of R with smallest number of leaves
8:         **for** each leaf X under C **do**
9:             $D \leftarrow$ `expand-then-merge`(N, D, X)
10:         mark C as processed
11: **return** ODD D

---

features outside R have been observed, and let C be a child of R. Observing all features under C leads to a new latent-tree classifier without the subtree rooted at C and an adjusted class prior. Algorithm 2 uses this observation by iteratively choosing a node C and then shrinking the classifier size by instantiating the features under C, allowing us to compile an ODD in a fashion similar to [Chan and Darwiche, 2003]. The specific choice of internal nodes C by Algorithm 2 leads to the following complexity.

Theorem 1 *Given a latent-tree classifier N with n variables, each with at most b values, the ODD computed by Algorithm 2 has size $O(b^{\frac{3n}{4}})$ and can be obtained in time $O(nb^{\frac{3n}{4}})$.*

If one makes further assumptions about the structure of the latent tree (e.g., if the root has k children, and each child of the root has $O(\frac{n}{k})$ features), then one obtains the size bound of $O(b^{\frac{n}{2}})$ and time bound of $O(nb^{\frac{n}{2}})$ for naive Bayes classifiers. We do not expect a significantly better upper bound on the time complexity due to the following result.

Theorem 2 *Given a naive Bayes classifier N, compiling an ODD representing its decision function is NP-hard.*

3 Minimum Cardinality Explanations

We now consider the first type of explanations for why a classifier makes a certain decision. These are called *minimum-cardinality explanations* or MC-explanations. We will first assume that the features are binary and then generalize later.

Consider two instances $\mathbf{x}^\star$ and $\mathbf{x}$. As we did earlier, we write $\mathbf{x}^\star \subseteq^1 \mathbf{x}$ to mean: the features set to 1 in $\mathbf{x}^\star$ are a subset of those set to 1 in $\mathbf{x}$. We define $\mathbf{x}^\star \subseteq^0 \mathbf{x}$ analogously. Moreover, we write $\mathbf{x} \leq^1 \mathbf{x}^\star$ to mean: the count of 1-features in $\mathbf{x}$ is no greater than their count in $\mathbf{x}^\star$. We define $\mathbf{x} \leq^0 \mathbf{x}^\star$ analogously.

Definition 3 (MC-Explanation) *Let $f(\mathbf{X})$ be a given decision function. An <u>MC-explanation</u> of a <u>positive</u> instance $\mathbf{x}$ is another positive instance $\mathbf{x}^\star$ such that $\mathbf{x}^\star \subseteq^1 \mathbf{x}$ and there is no other positive instance $\mathbf{x}' \subseteq^1 \mathbf{x}$ where $\mathbf{x}' <^1 \mathbf{x}^\star$. An <u>MC-explanation</u> of a <u>negative</u> instance $\mathbf{x}$ is another negative instance $\mathbf{x}^\star$ such that $\mathbf{x}^\star \subseteq^0 \mathbf{x}$ and there is no other negative instance $\mathbf{x}' \subseteq^0 \mathbf{x}$ where $\mathbf{x}' <^0 \mathbf{x}^\star$.*

Intuitively, an MC-explanation of a positive decision $f(\mathbf{x}) = 1$ answers the question: which positive features of instance $\mathbf{x}$ are responsible for this decision? Similarly for the MC-explanation of a negative decision $f(\mathbf{x}) = 0$: which negative features of instance $\mathbf{x}$ are responsible for this decision? MC-explanations are not necessarily unique as we shall see later. However, MC-explanations of positive decisions must all have the same number of 1-features, and those for negative decisions must all have the same number of 0-features.

MC-explanations are perhaps best illustrated using a monotone classifier. As a running example, consider a (monotone) classifier for deciding whether a student will be admitted to a university. The class variable is admit (A) and the features of an applicant are:

- work-experience (W): has prior work experience.
- first-time-applicant (F): did not apply before.
- entrance-exam (E): passed the entrance exam.
- gpa (G): has met the university's expected GPA.

All variables are either positive (+) or negative (-).

Consider a naive Bayes classifier with the following false positive and false negative rates:

feature	f_p	f_n
W	0.10	0.04
F	0.20	0.30
E	0.15	0.60
G	0.11	0.03

To completely specify the naive Bayes classifier, we also need the prior probability of admission, which we assume to be $Pr(A=+) = 0.30$. Moreover, we use a decision threshold of 0.50, admitting an applicant $\mathbf{x}$ if $Pr(A=+ \mid \mathbf{x}) \geq .50$. Note that with the above false positive and false negative rates, a positively observed feature will increase the probability of a positive classification, while a negatively observed feature will increase the probability of a negative classification (hence, the classifier is monotone).

Table 1 depicts the decision function f for this naive Bayes classifier, with MC-explanations for all 16 instances.

Consider, for example, a student (+ + + +) who was admitted by this decision function. There is a single MC-explanation for this decision, (+ - - +), with cardinality 2. According to this explanation, work experience and a good GPA were the reasons for admission. That is, the student would

W	F	E	G	$Pr(A{=}+\mid\mathbf{x})$	$f(\mathbf{x})$	MC-explanations
-	-	-	-	0.0002	-	(- - + +) (- + - +) (- + + -) (+ - + -) (+ + - -)
-	-	-	+	0.0426	-	(- - + +) (- + - +)
-	-	+	-	0.0006	-	(- - + +) (- + + -) (+ - + -)
-	-	+	+	0.1438	-	(- - + +)
-	+	-	-	0.0016	-	(- + - +) (- + + -) (+ + - -)
-	+	-	+	0.2933	-	(- + - +)
-	+	+	-	0.0060	-	(- + + -)
-	+	+	+	0.6105	+	(- + + +)
+	-	-	-	0.0354	-	(+ + - -) (+ - + -)
+	-	-	+	0.9057	+	(+ - - +)
+	-	+	-	0.1218	-	(+ - + -)
+	-	+	+	0.9732	+	(+ - - +)
+	+	-	-	0.2552	-	(+ + - -)
+	+	-	+	0.9890	+	(+ - - +)
+	+	+	-	0.5642	+	(+ + + -)
+	+	+	+	0.9971	+	(+ - - +)

Table 1: A decision function with MC-explanations.

still have been admitted even if they have applied before and did not pass the entrance exam.

For another example, consider a student (- - - +) who was rejected. There are two MC-explanations for this decision. The first, (- - + +), says that the student would not have been admitted, even if they passed the entrance exam. The second explanation, (- + - +), says that the student would not have been admitted, even if they were a first-time applicant.

Finally, we remark that while MC-explanations are more intuitive for monotone classifiers, they also apply to classifiers that are not monotone, as we shall see in Section 3.2.

3.1 Computing MC-Explanations

We will now present an efficient algorithm for computing the MC-explanations of a decision, assuming that the decision function has a specific form. Our treatment assumes that the decision function is represented as an OBDD, but it actually applies to a broader class of representations which includes OBDDs as a special case. More on this later.

Our algorithm uses a key operation on decision functions.

Definition 4 (Cardinality Minimization) *For $i \in \{0, 1\}$, the <u>i-minimization</u> of decision function $f(\mathbf{X})$ is another decision function $f^i(\mathbf{X})$ defined as follows: $f^i(\mathbf{x}) = 1$ iff (a) $f(\mathbf{x}) = 1$ and (b) $\mathbf{x} \leq^i \mathbf{x}^\star$ for every $f(\mathbf{x}^\star) = 1$.*

The 1-minimization of decision function f renders positive decisions only on the positive instances of f having a minimal number of 1-features. Similarly, the 0-minimization of decision function f renders positive decisions only on the positive instances of f having a minimal number of 0-features. Cardinality minimization was discussed and employed for other purposes in [Darwiche, 2001; Choi *et al.*, 2013].

Algorithm 3 computes the MC-explanations of a decision $f(\mathbf{x})$. The set of computed explanations is encoded by another decision function $g(\mathbf{X})$. In particular, $g(\mathbf{x}^\star) = 1$ iff $\mathbf{x}^\star$ is an MC-explanation of decision $f(\mathbf{x})$.

Suppose we want to compute the MC-explanations of a positive decision $f(\mathbf{x}) = 1$. The algorithm will first find the portion α of instance $\mathbf{x}$ with variables set to 0. It will then

Algorithm 3 `find-mc-explanation`$(f(\mathbf{X}), \mathbf{x})$

input: An OBDD $f(\mathbf{X})$ and instance $\mathbf{x}$.

output: An OBDD $g(\mathbf{X})$ where $g(\mathbf{x}^\star) = 1$ iff $\mathbf{x}^\star$ is an MC-explanation of decision $f(\mathbf{x})$.

main:

1: $i \leftarrow f(\mathbf{x})$
2: $\alpha \leftarrow$ the subset of $\mathbf{x}$ with variables set to $1 - i$
3: complement function f if $i = 0$
4: **return** i-minimize(conjoin(f, α))

---

conjoin[4] f with α and 1-minimize the result. The obtained decision function encodes the MC-explanations in this case.

An OBDD can be complemented and conjoined with a variable instantiation in linear time. It can also be minimized in linear time. This leads to the following complexity for generating MC-explanations based on OBDDs.

Theorem 3 *When the decision function $f(\mathbf{X})$ is represented as an OBDD, the time and space complexity of Algorithm 3 is linear in the size of f, while guaranteeing that the output function $g(\mathbf{X})$ is also an OBDD.*

Given OBDD properties, one can count MC-explanations in linear time, and enumerate each in linear time.[5]

3.2 Case Study: Votes Classifier

We now consider the Congressional Voting Records (votes) from the UCI machine learning repository [Bache and Lichman, 2013]. This dataset consists of 16 key votes by Congressmen of the U.S. House of Representatives. The class label is the party of the Congressman (positive if Republican and negative if Democrat). A naive Bayes classifier trained on this dataset obtains 91.0% accuracy. We compiled this classifier into an OBDD, which has a size of 630 nodes.

The following Congressman from the dataset voted on all 16 issues and was classified correctly as a Republican:

$$(0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1)$$

This decision has five MC-explanations of cardinality 3, e.g.:

$$(0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0)$$

The MC-explanation tells us that this Congressmen could have reversed four of their yes-votes, and the classifier would still predict that this Congressman was a Republican.

---

[4]Conjoining f with α leads to a function h such that $h(\mathbf{x}) = 1$ iff $f(\mathbf{x}) = 1$ and $\mathbf{x}$ is compatible with α.

[5]Minimization, conjoin, and model enumeration are all linear time operations on DNNFs, which is a superset of OBDDs [Darwiche, 2001; Darwiche and Marquis, 2002]. Moreover,

$$\text{OBDD} \subset \text{SDD} \subset \text{d-DNNF} \subset \text{DNNF}$$

where we read $\subset$ as "is-a-subclass-of". Hence, DNNFs, d-DNNFs and SDDs could have been used for supporting MC-explanations, except that we would need a different algorithm for compiling classifiers. Moreover, beyond OBDDs, only SDDs support complementation in linear time. Hence, efficiently computing MC-explanations of negative decision requires that we efficiently complement the decision functions represented by DNNFs or d-DNNFs.

W	F	E	G	$Pr(A{=}{+}|\mathbf{x})$	$f(\mathbf{x})$	PI-explanations
-	-	-	-	0.0002	-	$(\bar{w}\bar{f})\,(\bar{w}\bar{e})\,(\bar{w}\bar{g})\,(\bar{f}\bar{g})\,(\bar{e}\bar{g})$
-	-	-	+	0.0426	-	$(\bar{w}\bar{f})\,(\bar{w}\bar{e})$
-	-	+	-	0.0006	-	$(\bar{w}\bar{f})\,(\bar{w}\bar{g})\,(\bar{f}\bar{g})$
-	-	+	+	0.1438	-	$(\bar{w}\bar{f})$
-	+	-	-	0.0016	-	$(\bar{w}\bar{e})\,(\bar{w}\bar{g})\,(\bar{e}\bar{g})$
-	+	-	+	0.2933	-	$(\bar{w}\bar{e})$
-	+	+	-	0.0060	-	$(\bar{w}\bar{g})$
-	+	+	+	0.6105	+	(feg)
+	-	-	-	0.0354	-	$(\bar{f}\bar{g})\,(\bar{e}\bar{g})$
+	-	-	+	0.9057	+	(wg)
+	-	+	-	0.1218	-	$(\bar{f}\bar{g})$
+	-	+	+	0.9732	+	(wg)
+	+	-	-	0.2552	-	$(\bar{e}\bar{g})$
+	+	-	+	0.9890	+	(wg)
+	+	+	-	0.5642	+	(wfe)
+	+	+	+	0.9971	+	$(wg)\,(wfe)\,(feg)$

Table 2: A decision function with PI-explanations.

For a problem of this size, we can enumerate all instances of the classifier. We computed the MC-explanations for each of the $32,256$ positive instances, out of a possible number of $2^{16} = 65,536$ instances. Among these MC-explanations, the one that appeared the most frequently was the MC-explanation from the above example. This explanation corresponded to yes-votes on three issues: physician-fee-freeze, el-salvador-aid, and crime. Further examination of the dataset revealed that these issues were the three with the fewest Republican no-votes.

4 Prime Implicant Explanations

We now consider the second type of explanations, called *prime-implicant explanations* or PI-explanations for short.

Let $\mathbf{y}$ and $\mathbf{z}$ be instantiations of some features and call them *partial instances*. We will write $\mathbf{y} \supseteq \mathbf{z}$ to mean that $\mathbf{y}$ extends $\mathbf{z}$, that is, it includes $\mathbf{z}$ but may set some additional features.

Definition 5 (PI-Explanation) *Let $f(\mathbf{X})$ be a given decision function. A PI-explanation of a decision $f(\mathbf{x})$ is a partial instance $\mathbf{z}$ such that*

(a) $\mathbf{z} \subseteq \mathbf{x}$,

(b) $f(\mathbf{x}) = f(\mathbf{x}^\star)$ for every $\mathbf{x}^\star \supseteq \mathbf{z}$, and

(c) no other partial instance $\mathbf{y} \subset \mathbf{z}$ satisfies (a) and (b).

Intuitively, a PI-explanation of decision $f(\mathbf{x})$ is a minimal subset $\mathbf{z}$ of instance $\mathbf{x}$ that makes features outside $\mathbf{z}$ irrelevant to the decision. That is, we can toggle any feature that does not appear in $\mathbf{z}$ while maintaining the current decision. The number of features appearing in a PI-explanation will be called the *length* of the explanation. As we shall see later, PI-explanations of the same decision may have different lengths.

Table 2 depicts the decision function f for the admissions classifier, with PI-explanations for all 16 instances. We write (wg) for $W{=}{+}, G{=}{+}$ and $(\bar{e}\bar{g})$ for $E{=}{-}, G{=}{-}$.

Consider a student (+ + - -) who was *not* admitted by this decision function. There is a single PI-explanation $(\bar{e}\bar{g})$ for this decision. According to this explanation, it is sufficient to have a poor entrance exam and a poor GPA to be rejected—it

---

Algorithm 4 `pi-cover`(f, π)

---

input: OBDD f and variable ordering π

output: ODD g encoding prime implicants of f

main:

1: **if** π is empty **return** f
2: remove first variable X from order π
3: $g_* \leftarrow$ `pi-cover`$(f_{\bar{x}} \wedge f_x, \pi)$
4: $g_{\bar{x}} \leftarrow$ `pi-cover`$(f_{\bar{x}}, \pi)$, $g_x \leftarrow$ `pi-cover`(f_x, π)
5: $g_{\bar{x}} \leftarrow g_{\bar{x}} \wedge \neg g_*$, $g_x \leftarrow g_x \wedge \neg g_*$
6: **return** ODD with branches $g_{\bar{x}}, g_x, g_*$

---

does not matter whether they have work experience or if they are a first-time applicant. That is, we can set these features to any value, and the applicant would still be rejected.

Consider now a student $(+ + + +)$ who was admitted. There are three PI-explanations for this decision, (wg) (wfe) (feg), with different lengths. These explanations can be visualized as $(+ * * +)$, $(+ + + *)$ and $(* + + +)$. This is in contrast to the single MC-explanation $(+ - - +)$ obtained previously.

4.1 Computing Prime Implicant Explanations

Algorithms exist for converting an OBDD for function f into an ODD that encodes the prime implicants of f [Coudert and Madre, 1993; Coudert *et al.*, 1993; Minato, 1993].[6] The resulting ODD has three values for each variable: 0, 1 and $*$ (don't care). The ODD encodes partial instances, which correspond to the PI-explanations of positive instances (to get the PI-explanations of negative instances, we complement the OBDD f). These algorithms recurse on the structure of the input OBDD, computing prime implicants of sub-OBDDs. If X is the variable labeling the root of OBDD f, then $f_{\bar{x}}$ denotes its 0-child and f_x denotes its 1-child. Algorithm 4 computes prime implicants by recursively computing prime implicants for $f_{\bar{x}}$, f_x and $f_{\bar{x}} \wedge f_x$ [Coudert and Madre, 1993].

As we are interested in explaining a specific instance $\mathbf{x}$, we only need the prime implicants compatible with $\mathbf{x}$ (a function may have exponentially many prime implicants, but those compatible with an instance may be small). We exploit this observation in Algorithm 5, which computes the PI-explanations of a given positive instance $\mathbf{x}$ by avoiding certain recursive calls. Empirically, we have observed that Algorithm 5 can be twice as fast as Algorithm 4 (computing PIs first, then conjoining with a given instance to obtain PI-explanations). It can also generate ODDs that are an order-of-magnitude smaller. The following table highlights this difference in size and running time, per instance, between Algorithms 4 (cover) & 5 (inst). Relative improvements are denoted by impr; n denotes the number of features. We report averages over 50 instances.

dataset	n	time (s)			ODD size		
		cover	inst	impr	cover	inst	impr
votes	16	0.04	0.02	1.99	2,144	139	15.42
spect	22	0.06	0.02	2.27	3,130	437	7.14
msnbc	16	0.07	0.02	2.56	5,086	446	11.39
nltcs	15	0.03	0.02	1.39	432	111	3.89

[6]These algorithms compute prime-implicant *covers*.

Algorithm 5 `pi-inst`$(f, \pi, \mathbf{x})$

---

input: OBDD f, variable ordering π, and instance $\mathbf{x}$

output: ODD g for primes implicant compatible with $\mathbf{x}$

main:

1: **if** π is empty **return** f
2: remove first variable X from order π
3: $g_* \leftarrow$ `pi-inst`$(f_{\bar{x}} \wedge f_x, \pi, \mathbf{x})$
4: **if** $\mathbf{x}$ sets X to $\bar{x}$ **then**
5: $\quad g_{\bar{x}} \leftarrow$ `pi-inst`$(f_{\bar{x}}, \pi, \mathbf{x})$, $g_x \leftarrow \perp$
6: **else**
7: $\quad g_{\bar{x}} \leftarrow \perp$, $g_x \leftarrow$ `pi-inst`$(f_x, \pi, \mathbf{x})$
8: $g_{\bar{x}} \leftarrow g_{\bar{x}} \wedge \neg g_*$, $g_x \leftarrow g_x \wedge \neg g_*$
9: **return** ODD with branches $g_{\bar{x}}, g_x, g_*$

4.2 Case Study: Votes Classifier

Consider again the voting record of the Republican Congressman that we considered earlier in Section 3.2:

$$(0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1)$$

There are 30 PI-explanations of this decision. There are 2 shortest explanations of 9 features:

$$(\quad 0 \ 1 \ 1 \quad 0 \ 0 \ 0 \quad \quad 1 \ 1 \ 0 \quad)$$
$$(\quad 0 \ 1 \ 1 \ 1 \quad 0 \ 0 \quad \quad 1 \ 1 \ 0 \quad)$$

The first corresponds to yes votes on:

physician-fee-freeze, el-salvador-aid,
superfund-right-to-sue, crime,

and no votes on

adoption-of-the-budget-resolution, anti-satellite-test-ban,
aid-to-nicaraguan-contras, mx-missile, duty-free-exports.

These 9 votes necessitate the classification of a Republican; no other vote changes this decision. Finally, there are 506 PI-explanations for all decisions made by this classifier:

length of explanation	9	10	11	12	13	total
number of explanations	35	308	143	19	1	506

5 More On Monotone Classifiers

We now discuss a specific relationship between MC and PI explanations for monotone classifiers.

An MC-explanation sets all features, while a PI-explanation sets only a subset of the features. For a positive instance, we will say that MC-explanation $\mathbf{x}$ and PI-explanation $\mathbf{z}$ *match* iff $\mathbf{x}$ can be obtained from $\mathbf{z}$ by setting all missing features negatively. For a negative instance, MC-explanation $\mathbf{x}$ and PI-explanation $\mathbf{z}$ match iff $\mathbf{x}$ can be obtained from $\mathbf{z}$ by setting all missing features positively.

Theorem 4 *For a decision $f(\mathbf{x})$ of a monotone decision function f:*

1. *Each MC-explanation matches some shortest PI-explanation.*

2. *Each shortest PI-explanation matches some MC-explanation.*

Hence, for monotone decision functions, MC-explanations coincide with shortest PI-explanations.

The admissions classifier we considered earlier is monotone, which can be verified by inspecting its decision function (in contrast, the votes classifier is not monotone). Here, all MC-explanations matched PI-explanations. For example, the MC-explanation (+ - - +) for instance (+ + - +) matches the PI-explanation (wg). However, the PI-explanation (wfe) for instance (+ + + +) does not match the single MC-explanation (+ - - +). One can verify though, by examining Tables 1 and 2, that shortest PI-explanations coincide with MC-explanations.

MC-explanations are no longer than PI-explanations and their count is no larger than the count of PI-explanations. Moreover, MC-explanations can be computed in linear time, given that the decision function is represented as an OBDD. This is not guaranteed for PI-explanations.

PI-explanations can be directly extended to classifiers with multi-valued features. They are also meaningful for arbitrary classifiers, not just monotone ones. While our definition of MC-explanations was directed towards monotone classifiers with binary features, it can be generalized so it remains useful for arbitrary classifiers with multi-valued features. In particular, let us partition the *values* of each feature into two sets: on-values and off-values. Let us also partition the set of *features* **X** into **Y** and **Z**. Consider now the following question about a decision $f(\mathbf{x})$, where $\mathbf{x} = \mathbf{yz}$. Keeping **y** fixed, find a culprit of on-features in **z** that maintains the current decision. Definition 3 is a special case of this more general definition, and Algorithm 3 can be easily extended to compute these more general MC-explanations using the same complexity (that is, linear in the size of ODD for the decision function).

6 Related Work

There has been significant interest recently in providing explanations for classifiers; see, e.g., [Ribeiro *et al.*, 2016b; Elenberg *et al.*, 2017; Lundberg and Lee, 2017; Ribeiro *et al.*, 2016a; 2018]. In particular, *model-agnostic* explainers were sought [Ribeiro *et al.*, 2016b], which can explain the behavior of (most) any classifier, by treating it as a *black box*. Take for example, LIME, which *locally* explains the classification of a given instance. Roughly, LIME samples new instances that are "close" to a given instance, and then learns a simpler, interpretable model from the sampled data. For example, suppose a classifier rejects a loan to an applicant; one could learn a decision tree for other instances similar to the applicant, to understand why the original decision was made.

More related to our work is the notion of an "anchor" introduced in [Ribeiro *et al.*, 2016a; 2018]. An anchor for an instance is a subset of the instance that is highly likely to be classified with the same label, no matter how the missing features are filled in (according to some distribution). An anchor can be viewed as a probabilistic extension of a PI-explanation. Anchors can also be understood using the Same-Decision Probability (SDP) [Choi *et al.*, 2012; Chen *et al.*, 2014; Choi *et al.*, 2017], proposed in [Darwiche and Choi, 2010]. In this context, the SDP asks, "Given that I have already observed **x**, what is the probability that I will make the same classification if I observe the remaining fea-

tures?" In this case, we expect an anchor **x** to have a high SDP, but a PI-explanation **x** will always have an SDP of 1.0.

7 Conclusion

We proposed an algorithm for compiling latent-tree Bayesian network classifiers into decision functions in the form of ODDs. We also proposed two approaches for explaining the decision that a Bayesian network classifier makes on a given instance, which apply more generally to any decision function in symbolic form. One approach is based on MC-explanations, which minimize the number of positive features in an instance, while maintaining its classification. The other approach is based on PI-explanations, which identify a smallest set of features in an instance that renders the remaining features irrelevant to a classification. We proposed algorithms for computing these explanations when the decision function has a symbolic and tractable form. We also discussed monotone classifiers and showed that MC-explanations and PI-explanations coincide for this class of classifiers.

Acknowledgments

A Proofs

Proof of Theorem 1 Our proof is based on analyzing Algorithm 2 on an arbitrary latent-tree classifier with n variables and b values, and bounding the size of the decision graph D after each call to `expand-then-merge`. For any iteration of the while-loop, let D be the initial decision graph and let D' be the decision graph generated after the expanding phase of `expand-then-merge`$(., D, .)$. Furthermore, let $S(D)$ denote the number of leaf nodes of D (similarly for D'). We will show the following loop invariant: $S(D') \leq b^{\frac{3n}{4}}$. For any iteration, $S(D)$ is bounded by $\min(b^i, b^{n-i})$, where i denotes the depth of D. There are $n - i$ variables remaining, and the choice of C in the algorithm guarantees that the number of variables under C is at most $\frac{n-i}{2}$. Thus, $S(D')$ is bounded by $b^{\frac{n-i}{2}} S(D) = \min(b^{\frac{n+i}{2}}, b^{\frac{3(n-i)}{2}})$. If $i \leq \frac{n}{2}$, then $S(D') \leq b^{\frac{n+n/2}{2}} = b^{\frac{3n}{4}}$. Otherwise if $i > \frac{n}{2}$ then $S(D') \leq b^{\frac{3(n-n/2)}{2}} = b^{\frac{3n}{4}}$. Thus, after every call to `expand-then-merge`, the decision graph D' has at most $b^{\frac{3n}{4}}$ leaf nodes and the merging phase cannot increase the number of nodes, giving us a total size bound of $O(nb^{\frac{3n}{4}})$. To obtain the size bound of $O(b^{\frac{3n}{4}})$, observe that $S(D')$ is at least half of the number of newly expanded nodes for each call, and at most one such call can have $S(D') > b^{\frac{2n}{3}}$ nodes. Finally, merging a node in D' takes time logarithmic in the size of D', so the time complexity is $O(nb^{\frac{3n}{4}})$. □

Proof of Theorem 2 Our proof is based on [Chen *et al.*, 2014], which showed that computing the same-decision probability (SDP) is NP-hard in naive Bayes networks. Say we have an instance of the number partitioning problem, where we have positive integers $a_1, \ldots, a_n$ and we ask if there exists

a set $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$. Suppose we have a naive Bayes classifier with features X_i where:

$$\log \frac{Pr(x_i \mid c)}{Pr(x_i \mid \bar{c})} = a_i \quad \text{and} \quad \log \frac{Pr(\bar{x}_i \mid c)}{Pr(\bar{x}_i \mid \bar{c})} = -a_i$$

and where we have a uniform prior $Pr(C)$. Let $\mathbf{x}_I$ be the instance where X_i is set to true if $i \in I$ and X_i is set to false if $i \notin I$. Consider the log-odds $\log O(c \mid \mathbf{x}_I) = \log \frac{Pr(c \mid \mathbf{x}_I)}{Pr(\bar{c} \mid \mathbf{x}_I)}$:

$$\log O(c \mid \mathbf{x}_I) = \sum_{i \in I} \log \frac{Pr(x_i \mid c)}{Pr(x_i \mid \bar{c})} + \sum_{i \notin I} \log \frac{Pr(\bar{x}_i \mid c)}{Pr(\bar{x}_i \mid \bar{c})}$$

$$= \left(\sum_{i \in I} a_i \right) - \left(\sum_{i \notin I} a_i \right)$$

If I is a number partitioning solution, then $\log O(c \mid \mathbf{x}_I) = 0$. Otherwise $\log O(c \mid \mathbf{x}_I) = -\log O(c \mid \mathbf{x}_J) \neq 0$ where $J = \{1, \ldots, n\} \setminus I$. Hence, if there is no solution I, then half of the instances $\mathbf{x}$ have log-odds strictly greater than zero, and the other half have log-odds strictly less than zero. Thus, there exists a solution iff the number of positive instances in the decision function of N is strictly less than $\frac{1}{2} \cdot 2^n$ given a (strict) threshold of $\frac{1}{2}$. Finally, if we can compile the decision function of N to an OBDD in polytime, then we can perform model counting in time linear in the size of the OBDD, and hence solve number partitioning, which is NP-complete. Thus, compiling the decision function is NP-hard. $\square$

Proof of Theorem 3 An OBDD f can be complemented by simply switching its 0-sink and 1-sink. Since α is a conjunction of literals, we can conjoin f with α by manipulating the OBDD structure directly: if X appears in α positively (negatively), we redirect the 0-edge (1-edge) of each OBDD node labeled by X to the 0-sink. Clearly, this operation takes time linear in the size of f. The operation of i-minimization can also be performed in time linear in the size of f using the technique given in [Darwiche, 2001] for DNNFs. The minimization procedure performs two passes. The first pass performs an addition or minimization at each node. The second pass redirects some edges depending on simple tests. $\square$

Proof of Theorem 4 Suppose, without loss of generality, that we are explaining a positive instance $\mathbf{x}^\star$ of a monotone decision function f (the negative case is symmetric). The proof uses the following observation: A shortest PI-explanation $\mathbf{z}$ must have all its features set positively (otherwise, due to monotonicity, we can just drop the negative features in $\mathbf{z}$ to obtain a shorter PI-explanation).

1. Suppose that $\mathbf{x}$ is an MC-explanation. Let $\mathbf{z}$ be the portion of $\mathbf{x}$ containing all features that are set positively. Due to monotonicity, we can toggle features of $\mathbf{x}$ that are outside $\mathbf{Z}$ without changing the decision. Moreover, no subset of $\mathbf{z}$ will have this property; otherwise, $\mathbf{x}$ cannot be an MC-explanation. Hence, $\mathbf{z}$ is a PI-explanation that matches $\mathbf{x}$. Suppose now that $\mathbf{z}$ is not a shortest PI-explanation and let $\mathbf{z}'$ be a shortest PI-explanation. Then we can augment $\mathbf{z}'$ by setting all missing features negatively, giving us a positive instance with a 1-cardinality less than that of $\mathbf{x}$. Hence, $\mathbf{x}$ cannot be an MC-explanation.

2. Suppose that $\mathbf{z}$ is a shortest PI-explanation. Then all features in $\mathbf{z}$ must be set positively. Now let $\mathbf{x}$ be the result of augmenting $\mathbf{z}$ by setting all missing features negatively. Then $\mathbf{x}$ is a positive instance since $\mathbf{z}$ is a PI-explanation. Suppose now that $\mathbf{x}$ is not an MC-explanation, and let $\mathbf{x}'$ be an MC-explanation. Then let $\mathbf{z}'$ be the portion of $\mathbf{x}'$ containing all features that are set positively. By monotonicity, $\mathbf{z}$ cannot be a shortest PI-explanation since $\mathbf{z}'$ is shorter than $\mathbf{z}$ yet all of its completions would be positive instances.

References

[Bache and Lichman, 2013] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[Bryant, 1986] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.

[Chan and Darwiche, 2003] Hei Chan and Adnan Darwiche. Reasoning about Bayesian network classifiers. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 107–115, 2003.

[Chen et al., 2014] Suming Chen, Arthur Choi, and Adnan Darwiche. Algorithms and applications for the same-decision probability. *Journal of Artificial Intelligence Research (JAIR)*, 49:601–633, 2014.

[Choi et al., 2012] Arthur Choi, Yexiang Xue, and Adnan Darwiche. Same-decision probability: A confidence measure for threshold-based decisions. *International Journal of Approximate Reasoning (IJAR)*, 53(9):1415–1428, 2012.

[Choi et al., 2013] Arthur Choi, Doga Kisa, and Adnan Darwiche. Compiling probabilistic graphical models using sentential decision diagrams. In *Proceedings of the 12th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 121–132, 2013.

[Choi et al., 2017] YooJung Choi, Adnan Darwiche, and Guy Van den Broeck. Optimal feature selection for decision robustness in Bayesian networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1554–1560, August 2017.

[Coudert and Madre, 1993] Olivier Coudert and Jean Christophe Madre. Fault tree analysis: 10^{20} prime implicants and beyond. In *Proc. of the Annual Reliability and Maintainability Symposium*, pages 240–245, 1993.

[Coudert et al., 1993] Olivier Coudert, Jean Christophe Madre, Henri Fraisse, and Herve Touati. Implicit prime cover computation: An overview. In *Proceedings of the 4th SASIMI Workshop*, 1993.

[Darwiche and Choi, 2010] Adnan Darwiche and Arthur Choi. Same-decision probability: A confidence measure for threshold-based decisions under noisy sensors. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM)*, pages 113–120, 2010.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research (JAIR)*, 17:229–264, 2002.

[Darwiche, 2001] Adnan Darwiche. Decomposable negation normal form. *Journal of the ACM*, 48(4):608–647, 2001.

[Elenberg *et al.*, 2017] Ethan R. Elenberg, Alexandros G. Dimakis, Moran Feldman, and Amin Karbasi. Streaming weak submodularity: Interpreting neural networks on the fly. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 4047–4057, 2017.

[Horiyama and Ibaraki, 2002] Takashi Horiyama and Toshihide Ibaraki. Ordered binary decision diagrams as knowledge-bases. *Artificial Intelligence (AIJ)*, 136(2):189–213, 2002.

[Lundberg and Lee, 2017] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 4768–4777, 2017.

[Meinel and Theobald, 1998] Christoph Meinel and Thorsten Theobald. *Algorithms and Data Structures in VLSI Design: OBDD — Foundations and Applications*. Springer, 1998.

[Minato, 1993] Shin-ichi Minato. Fast generation of prime-irredundant covers from binary decision diagrams. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 76(6):967–973, 1993.

[Ribeiro *et al.*, 2016a] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Nothing else matters: Model-agnostic explanations by identifying prediction invariance. In *NIPS Workshop on Interpretable Machine Learning in Complex Systems*, 2016.

[Ribeiro *et al.*, 2016b] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Knowledge Discovery and Data Mining (KDD)*, pages 1135–1144, 2016.

[Ribeiro *et al.*, 2018] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, pages 1527–1535, 2018.

[van der Gaag *et al.*, 2004] Linda C. van der Gaag, Hans L. Bodlaender, and A. J. Feelders. Monotonicity in Bayesian networks. In *Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 569–576, 2004.

[Wegener, 2000] Ingo Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, 2000.