Logic for Explainable AI

Adnan Darwiche
Computer Science Department
University of California
Los Angeles, USA
darwiche@cs.ucla.edu

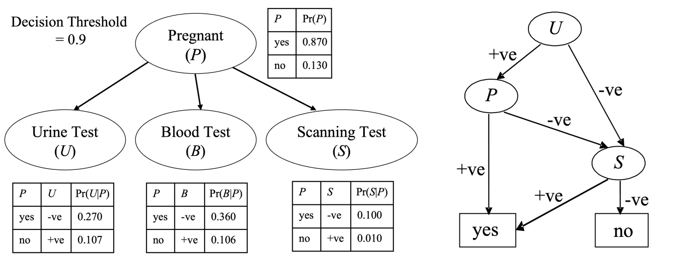
***Abstract*—A central quest in explainable AI relates to understanding the decisions made by (learned) classifiers. There are three dimensions of this understanding that have been receiving significant attention in recent years. The first dimension relates to characterizing conditions on instances that are necessary and sufficient for decisions, therefore providing abstractions of instances that can be viewed as the "reasons behind decisions." The next dimension relates to characterizing minimal conditions that are sufficient for a decision, therefore identifying maximal aspects of the instance that are irrelevant to the decision. The last dimension relates to characterizing minimal conditions that are necessary for a decision, therefore identifying minimal perturbations to the instance that yield alternate decisions. We discuss in this tutorial a comprehensive, semantical and computational theory of explainability along these dimensions which is based on some recent developments in symbolic logic. The tutorial will also discuss how this theory is particularly applicable to non-symbolic classifiers such as those based on Bayesian networks, decision trees, random forests and some types of neural networks.**

***Index Terms*—Explainable AI, symbolic logic, classifiers, prime implicants, prime implicants, quantified logic**

I. Introduction

Explaining the decisions of (learned) classifiers is perhaps the most studied task in the area of explainable AI. A classifier can take different forms—like a decision tree, random forest, Bayesian network or a neural network—but it is essentially a function that maps *instances* to a finite number of *classes.* When a classifier maps an instance to a class, we say it has made a *decision* on that instance. Each classifier has a set of *features* which are discrete or numeric variables. An instance is generated by assigning a value to each feature. Consider the Naïve Bayes classifier in Fig. 1(a) which has three binary features (U, B, S), representing medical tests, which generate eight instances. This classifier has two classes (P=YES, P=NO) corresponding to whether the patient is pregnant or not. Given an instance which represents the test results of a patient, this classifier first computes the distribution on variable P given these results and then assigns the class P=YES to the instance iff this probability is no less than 0.90 (called the classification threshold). This classifier could have been learned from data and it makes decisions by performing probabilistic reasoning, but it is in essence a function that maps a finite number of instances to classes. Fig. 2(a) depicts another classifier in the form of a decision tree. This one has two numeric features

This work has been partially supported by NSF grant #ISS-1910317.



(a) Naïve Bayes classifier (b) decision graph

Fig. 1. A Naïve Bayes classifier, from [1], with binary features U, B, S and classes P=YES, P=NO. The decision graph represents the same classifier as it will make the exact same decision on every instance.

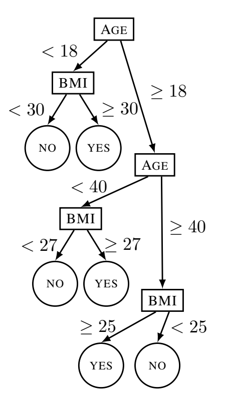
(AGE, BMI) and, hence, an infinite number of instances which are mapped by the classifier into one of two classes (YES, NO).

The goal of this tutorial is to discuss a theory based on symbolic logic for explaining the decisions of such classifiers. The theory targets three fundamental questions:

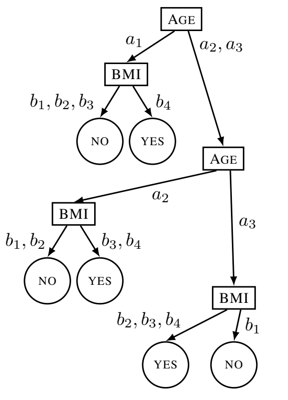
Q1. What is the reason for a decision on an instance?
Q2. What minimal aspects of an instance will guarantee the decision on that instance?
Q3. What minimal changes to an instance will lead to a different decision on that instance?

The answer to the first question will be a necessary and sufficient condition for the decision, expressed as a logical formula, which captures the essence of the instance that led to the decision. The answer to the second question will also be a logical formula. It will characterize minimal sets of features with weakest conditions on their states that are guaranteed to trigger the decision. This will immediately identify features that are irrelevant to the decision and will also identify irrelevant states of relevant features. The answer to the third question will also be a logical formula but one that identifies minimal sets of features and how they may be changed to flip the decision into some other, designated or arbitrary, decision. As we shall see later, the answer to the first question will form the basis for answering the second and third questions.

The discussed theory is based on representing classifiers using *class formulas* which are logical formulas that characterize the instances in each class. Two questions arise here. First, where do we get these class formulas from? Second, is it always possible to represent a classifier this way? The first question may arise when considering a classifier such



(a) decision tree



(b) discretized decision tree

Fig. 2. A decision tree, from [2], with numeric features AGE, BMI and classes YES, NO. AGE is discretized into three intervals $[0, 18)$, $[18, 40)$ and $[40, \infty)$ so it can be treated as a discrete variable with respective values a_1, a_2, a_3. BMI is discretized into four intervals $[0, 25)$, $[25, 27)$, $[27, 30)$, $[30, \infty)$ so it can be treated as a discrete variable with respective values b_1, b_2, b_3, b_4.

as the one in Fig. 1(a) which is numeric in nature and is based on probabilistic reasoning. The features of this classifier are discrete so the instances in each class are finite and can be represented by a logical formula that is obtained through a *compilation* process to be discussed at the end of this tutorial, in Section VI. The second question arises in the context of classifiers like the decision tree in Fig. 2(a) which involves an infinite number of instances due to the presence of numeric features. It is well known that such decision trees can be easily discretized as shown in Fig. 2(b). Here, the numeric values of the feature AGE are partitioned into three equivalence intervals (labeled a_1, a_2, a_3) and the ones for BMI are partitioned into four equivalence intervals (labeled b_1, b_2, b_3, b_4). Two instances whose features have point values in the same corresponding intervals are isomorphic from the viewpoint of this classifier, leading to a finite number of equivalence classes for the instances in each class so they can also be represented using logical formulas. The same applies to a broad set of classifiers including decision graphs and random forests with majority voting. Again, the compilation of class formulas for such classifiers will be discussed in Section VI.

We start this tutorial by some technical preliminaries in Section II and follow by addressing the three explainability questions in Sections III, IV and V. We will then discuss the compilation of classifiers into class formulas in Section VI and finally close with some remarks in Section VII.

II. DISCRETE LOGIC

It is quite common in the literature to employ Boolean logic even in the presence of discrete variables which are treated by "binarization." For example, a discrete variable with n values may be encoded using n Boolean variables; see [3] for a review of several encoding schemes. We will not follow this tradition in this tutorial. Instead, we will work with what one may call *discrete logic* as done in [2], [4]. In this logic, discrete variables are first class citizens that include Boolean variables as a special case. This is *very critical* semantically for the discussed theory of explainability and can be critical for computation too.[1] The bias towards Boolean logic is due to its potent and vast computational machinery, developed over decades, which does not apply directly to discrete variables. As we shall see, however, we now have potent tools for working with discrete variables in the context of explainable AI.

A. Syntax and Semantics of Discrete Logic

We assume a finite set of discrete variables Σ. Each variable $X \in \Sigma$ has a finite number of *states* $x_1, \ldots, x_n$, $n > 1$. A *literal* ℓ for variable X, called an X-literal, is a set of states such that $\emptyset \subset \ell \subset \{x_1, \ldots, x_n\}$. We often denote a literal such as $\{x_1, x_3, x_4\}$ by x_{134} which reads: the state of variable X is either x_1 or x_3 or x_4. A literal is *simple* iff it contains a single state. Hence, x_3 is a simple literal but x_{134} is not. Since a simple literal corresponds to a state, we use these two notions interchangeably. If a variable X has two states, it is said to be *Boolean* or *binary* and its states are denoted by x and $\overline{x}$. Such a variable can only have two simple literals, $\{x\}$ and $\{\overline{x}\}$, denoted x and $\overline{x}$ for convenience. A *formula* is either a constant $\top$, $\bot$, literal ℓ, negation $\overline{\alpha}$, conjunction $\alpha \cdot \beta$ or disjunction $\alpha + \beta$ where α, β are formulas. The set of variables appearing in a formula Δ are denoted by $\texttt{vars}(\Delta)$.

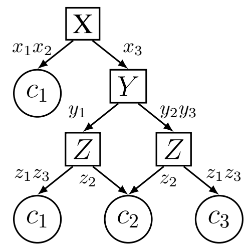
The semantics of discrete logic directly generalize those of Boolean logic. A *world,* denoted ω, maps each variable in Σ to one of its states. A world ω is called a *model* of formula α, written $\omega \models \alpha$, iff α is satisfied by ω (that is, α is true at ω). The constant $\top$ denotes a valid formula (satisfied by every world) and the constant $\bot$ denotes an unsatisfiable formula (has no models). Formula α implies/entails formula β, written $\alpha \models \beta$, iff every model of α is also a model of β. In this case, we say α is *stronger* than β and also say β is *weaker* than α.

B. Representing Classifiers using Class Formulas

A discrete variable $X \in \Sigma$ is called a *feature* and a state x_i (or simple literal $\{x_i\}$) is called a *characteristic.* An *instance* is a conjunction of characteristics, one for each feature in Σ (instances are in one-to-one correspondence with worlds).

Definition 1: A *classifier* with classes $c_1, \ldots, c_n$ is a set of mutually exclusive and exhaustive formulas $\Delta_1, \ldots, \Delta_n$, called *class formulas.* Instance $\mathcal{I}$ *is in class* c_i iff $\mathcal{I} \models \Delta_i$.

The decision graph below is a classifier with three ternary features X, Y, Z and three classes c_1, c_2, c_3. Its class formulas are $\Delta_1 = x_{12} + x_3 \cdot y_1 \cdot z_{13}$, $\Delta_2 = x_3 \cdot z_2$ and $\Delta_3 = x_3 \cdot y_{23} \cdot z_{13}$. This classifier has 27 instances: 20 in class c_1, 3 in class c_2 and 4 in class c_3. For example, instance $\mathcal{I} = x_3 \cdot y_2 \cdot z_2$ is in class c_2 since $\mathcal{I} \models \Delta_2$. Again, even though a classifier may take different forms, the presented theory of explainability views it as a set of class formulas since this is all we need to explain decisions.



[1]We have seen, for example, random forests whose discretization leads to variables with thousands of values. We want to avoid binarization in this case.

C. Explanations as Normal Forms

The answers to questions Q1, Q2 and Q3 will be expressed using logical formulas that have normal forms, discussed next. A *term* is a conjunction of literals for distinct variables. A *clause* is a disjunction of literals for distinct variables. A *Disjunctive Normal Form (DNF)* is a disjunction of terms. A *Conjunctive Normal Form (CNF)* is a conjunction of clauses. A *Negation Normal Form (NNF)* is a formula with no negations. These definitions imply that terms cannot be inconsistent, clauses cannot be valid, DNFs and CNFs are NNFs, and none of these normal forms can include negations. We say a formula (term/clause) is *simple* if it contains only simple literals. Simple and non-simple formulas will delineate two approaches for answering questions Q1, Q2 and Q3 based on the amount of information they convey about decisions.

D. Constructing Abstractions using Conditioning

When constructing abstractions of instances to explain decisions, the notion of conditioning plays a central role. Observe first that a simple term like $x_1 \cdot y_3 \cdot z_2$ represents a set of states. The *conditioning* of formula Δ on a simple term τ is denoted by $\Delta|\tau$ and obtained as follows. For each state x_i that appears in the simple term τ, replace each X-literal ℓ in Δ with $\top$ if $x_i \in \ell$; otherwise, replace ℓ with $\bot$. Consider the formula $\Delta = x_{12} + x_3 \cdot y_1 \cdot z_{13}$ and the simple term $\tau = x_3 \cdot z_1$. Then $\Delta|\tau = \bot + \top \cdot y_1 \cdot \top = y_1$. In general, the conditioned formula $\Delta|\tau$ does not mention any variable that appears in term τ.

E. Minimality using Prime Implicants and Implicates

Questions Q2 and Q3 involve a notion of minimality that will be captured using the classical notions of prime implicants and implicates. A term τ is called an *implicant* of formula Δ iff it implies Δ ($\tau \models \Delta$). It is called a *prime implicant* if no other implicant of Δ is weaker than τ (i.e., $\tau \models \tau' \models \Delta$ does not hold for any term $\tau' \neq \tau$). A clause σ is an *implicate* of formula Δ iff it is implied by Δ ($\Delta \models \sigma$). It is called a *prime implicate* if no other implicate of Δ is stronger than σ (i.e., $\Delta \models \sigma' \models \sigma$ does not hold for any clause $\sigma' \neq \sigma$). A prime implicant τ is *variable-minimal* iff no other prime implicant τ' satisfies $\texttt{vars}(\tau') \subset \texttt{vars}(\tau)$. Variable-minimal prime implicates are defined similarly.

The prime implicants and implicates of discrete formulas behave in ways that may surprise someone who is accustomed to these notions in a Boolean setting. Moreover, these behaviors have interesting implications on questions Q2 and Q3. Consider the Boolean formula $\Delta_b = (x + \overline{z}) \cdot (x \cdot z + y)$. The term $x \cdot y \cdot \overline{z}$ is an implicant but not prime. The only way to weaken this term so it becomes prime is by dropping some of its variables ($y \cdot \overline{z}$ is a prime implicant). In discrete logic, we can possibly weaken a term without dropping any of its variables, by adding states to some of its literals. Consider the discrete formula $\Delta_d = (x \cdot z_1 + z_2) \cdot (x \cdot z_3 + y)$, where variable Z is ternary and variables X, Y are binary. The term $x \cdot y \cdot z_1$ is an implicant of this formula but is not prime. We can weaken this term by adding the state z_2 to literal z_1, leading to the prime implicant $x \cdot y \cdot z_{12}$. A symmetrical situation arises for prime implicates: the only way to strengthen a Boolean clause is by dropping some of its variables but we can strengthen a discrete clause by removing states from its literals.

III. The Reasons Behind Decisions

We now turn to answering question Q1: What is the reason for a decision on an instance? We will present two different answers to this question. The first underlies extensive developments in explainable AI over the last few years. The second is very recent, more informative and further subsumes the first answer. The two answers, however, are equivalent if all features of the classifier are binary (i.e., Boolean).

Consider a classifier specified by class formulas $\Delta_1, \ldots, \Delta_n$ and suppose it decides that instance $\mathcal{I}$ belongs to class c_i, $\mathcal{I} \models \Delta_i$. We need to know why. The answer to this question must be a condition on the instance that implies formula Δ_i. Since this condition is meant to represent an abstraction of the instance, we want it to be as weak as possible so we get the most general abstraction. Here where things get interesting. If we do not place additional requirements on this condition, we will get a trivial answer to question Q1 since the class formula Δ_i satisfies the requirements we stated. To see why, $\mathcal{I} \models \Delta_i$ by supposition so Δ_i is indeed a condition satisfied by the instance. Moreover, there is no other condition on instance $\mathcal{I}$ that is weaker than Δ_i and that implies Δ_i. This answer is trivial as it will be the same answer returned when explaining the decision on any instance in class c_i. Hence, we need an additional requirement on the sought condition and we next review two proposals that have been extended for this purpose.

A. Complete Reasons

The first proposal requires the sought condition—that is, the reason for the decision—to be constructed from states that appear in the instance which can be combined in any fashion but using only conjunctions and disjunctions. Technically speaking, it requires the condition to be a simple NNF formula whose literals all appear in the instance being explained. This leads to the notion of *complete reason* introduced in [5].[2]

Definition 2: Suppose instance $\mathcal{I}$ belongs to class c_i. The *complete reason* for the decision on instance $\mathcal{I}$ is the weakest NNF Γ whose literals are in $\mathcal{I}$ and that satisfies $\mathcal{I} \models \Gamma \models \Delta_i$.

We stress here that the complete reason is a simple NNF formula, that is, every literal it contains corresponds to a state.

Consider a classifier with three ternary features X, Y, Z. Instance $\mathcal{I} = x_2 \cdot y_2 \cdot z_1$ is decided as belonging to class c_2 which has formula $\Delta_2 = x_{23} \cdot (x_2 + y_{23}) \cdot (y_{23} + z_1)$. The complete reason for this decision is $x_2 \cdot (y_2 + z_1)$. It says that instance $\mathcal{I}$ belongs to class c_1 because it has characteristic x_2 and one of the two characteristics y_2, z_1. This is indeed the most general condition on the instance which implies the class formula and that is constructed by applying conjunctions and disjunctions to the instance characteristics x_2, y_2 and z_1.

Two questions become relevant now. First, how do we compute the complete reason? Second, how can we use complete

[2]Ref. [5] gave a different but equivalent definition and for binary variables. See [4] for a more general treatment.

reasons when they are too complex to be interpretable by humans? We will address the first question next while leaving the second question until later in the tutorial.

Ref. [5] showed that if the class formula is appropriately represented (e.g., as an OBDD [6]), the complete reason can be computed in linear time. A major development then came in [7] which introduced the following quantification operator and showed that it can be used to express the complete reason.

Definition 3: For variable X with states $x_1, \ldots, x_n$, the *universal literal quantification* of state x_i from formula Δ is defined as $\forall x_i \cdot \Delta = \Delta | x_i \cdot \prod_{j \neq i}(x_i + \Delta | x_j)$.

This operator is commutative so it is meaningful to quantify an instance $\mathcal{I}$ from its class formula Δ_i, written $\forall \mathcal{I} \cdot \Delta_i$, by quantifying the states of $\mathcal{I}$ in any order. Moreover, $\forall \mathcal{I} \cdot \Delta_i$ corresponds to the complete reason as shown in [7] for Boolean variables and in [4] for discrete variables. One significance of this result is that this quantification operator is well behaved computationally on a number of logical forms. For example, one can quantify any set of states from a CNF in linear time. More generally, the operator distributes over conjunctions, and distributes over disjunctions when the disjuncts do not share variables as also shown in [7] (see weaker conditions in [4]). This brought into focus the following form of class formulas as it allows one to compute complete reasons in linear time (and general reasons too that are discussed in Section III-B).

Definition 4: An NNF is *or-decomposable* iff $\texttt{vars}(\alpha) \cap \texttt{vars}(\beta) = \emptyset$ for any disjuncts α and β in the NNF.[3]

Due to this form and some of its weakenings, [4] provided closed-form complete reasons for certain classes of classifiers that are based on decision graphs.

B. General Reasons

We now turn to a recent development [2] that identified a weaker requirement on abstractions of instances which can produce more information when explaining decisions.[4]

Definition 5: Suppose instance $\mathcal{I}$ is in class c_i. The *general reason* of the decision on instance $\mathcal{I}$ is the weakest NNF Γ whose literals are implied by $\mathcal{I}$ and that satisfies $\mathcal{I} \models \Gamma \models \Delta_i$.

The only difference between the complete reason and the general reason is that the former requires literals to appear in the instance, while the latter only requires that they are implied by the instance. Hence, general reasons are no longer simple formulas like complete reasons as they may contain non-simple literals. If all features are binary, then every literal is simple and the complete and general reasons are equal. However, in the presence of non-binary features, general reasons provide more information about why a decision was made. Let us see this through an example.

Consider the classifier in Fig. 3 and a patient Rob,

$$(\textsc{Diabetes}{=}\textsc{y}) \cdot (\textsc{Weight}{=}\textsc{under}) \cdot (\textsc{Btype}{=}\textsc{A}),$$

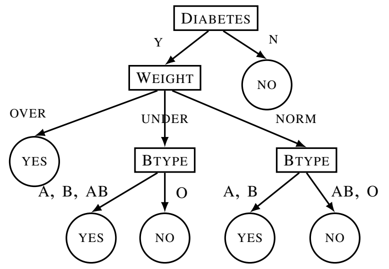


Fig. 3. A classifier of some disease in the form of a decision tree [2].

decided as YES. The complete reason for the decision is

$$(\textsc{Diabetes}{=}\textsc{y}) \cdot (\textsc{Btype}{=}\textsc{A})$$

and the general reason is

$$(\textsc{Diabetes}{=}\textsc{y}) \cdot (\textsc{Btype} \in \{\textsc{A, B, AB}\}) \cdot (\textsc{Weight} \in \{\textsc{under, over}\} + \textsc{Btype} \in \{\textsc{A, B}\}).$$

The complete reason justifies the decision by Rob having DIABETES and a BTYPE of A. But the general reason provides weaker justifications. One of them is that Rob has DIABETES and his BTYPE is not AB or O. Another justification is that he has DIABETES, his WEIGHT is not NORM and his BTYPE is not O.

The additional information provided by general reasons is particularly critical in the context of discretized classifiers. Consider again the decision tree in Fig. 2 which has two numeric features AGE, BMI. The feature AGE is discretized into three intervals $[0, 18)$, $[18, 40)$ and $[40, \infty)$ with corresponding labels a_1, a_2, a_3. The feature BMI is discretized into four intervals $[0, 25)$, $[25, 27)$, $[27, 30)$, $[30, \infty)$ with corresponding labels b_1, b_2, b_3, b_4. Hence, AGE can be treated as a discrete variable with states a_1, a_2, a_3 and BMI can be treated as a discrete variable with states b_1, b_2, b_3, b_4. Consider now the instance $\textsc{Age}{=}42 \cdot \textsc{bmi}{=}28$ which is discretized into $a_3 \cdot b_3$ ($\textsc{Age} \geq 40$ and $27 \leq \textsc{bmi} < 30$). The decision on this instance is YES and its class formula $\Delta_{\textsc{yes}}$ is $a_1 \cdot b_4 + a_2 \cdot b_{34} + a_3 \cdot b_{234}$ (obtained by tracing the paths leading into class YES). The complete reason for this decision is $a_3 \cdot b_3$ which is the instance itself so no abstraction took place. The general reason is $a_{23} \cdot b_{34} + a_3 \cdot b_{234}$ which reads

$$(\textsc{Age} \geq 18) \cdot (\textsc{bmi} \geq 27) + (\textsc{Age} \geq 40) \cdot (\textsc{bmi} \geq 25).$$

This is a weaker property of the instance that still implies the decision. For example, the first part says the decision is justified by AGE being no less than 18 and BMI being no less than 27. These kind of justifications are impossible to obtain using complete reasons as such reasons can only reference instance characteristics, that is, $\textsc{Age} \geq 40$ and $27 \leq \textsc{bmi} < 30$.

General reasons can also be expressed using a quantification operator introduced in [2].

Definition 6: For variable X with states $x_1, \ldots, x_n$ and formula Δ, $\overline{\forall} x_i \cdot \Delta$ is defined as $\Delta | x_i \cdot \Delta$.

This operator is also commutative so we can quantify an instance $\mathcal{I}$ from its class formula Δ_i, written $\overline{\forall} \mathcal{I} \cdot \Delta_i$, by quantifying the states of $\mathcal{I}$ in any order. Moreover, $\overline{\forall} \mathcal{I} \cdot \Delta_i$

[3]This is to be contrasted with and-decomposable NNFs, known as Decomposable NNFs (DNNFs) [8], which have been studied extensively.

[4]Def. 5 is actually a proposition in [2]. We reversed the formulation here to serve the storyline in this tutorial.

corresponds to the general reason for the decision on instance $\mathcal{I}$ as shown in [2]. The operators $\forall$ and $\overline{\forall}$ have similar computational properties as far as distributivity and tractability on forms such as CNFs and or-decomposable NNFs. This is why, similar to complete reasons, we also have closed-form general reasons for certain classes of classifiers [2].

C. Fixation and Monotonocity

Complete and general reasons satisfy a property called fixation, which has key implications including on computation as we discuss later. This property was also identified in [2].

Definition 7: An NNF is *locally fixated* on instance $\mathcal{I}$ iff its literals are consistent with $\mathcal{I}$. A formula is *fixated* on instance $\mathcal{I}$ iff it is equivalent to an NNF that is locally fixated on $\mathcal{I}$.

The formula $x_{12} \cdot y_{13} + z_{23}$ is locally fixated on the instance $\mathcal{I} = x_1 \cdot y_1 \cdot z_2$, but this is not true for the formula $x_{12} \cdot y_{23} + z_{23}$ since the literal y_{23} is not consistent with the instance $\mathcal{I}$. The fixation of complete reasons follows directly from Def. 2 and the fixation of general reasons follows directly from Def. 5.

If a simple formula is fixated then it is also monotone.

Definition 8: A formula is *monotone* iff for each variable X all X-literals that occur in the formula are simple and equal.

For example, the formula $(x_1 + y_2) \cdot (x_1 + z_3) \cdot (y_2 + z_3)$ is monotone, but the formulas $(x_1 + y_2) \cdot (x_{12} + z_3) \cdot (y_2 + z_3)$ is not monotone since it contains a non-simple literal x_{12}. The formula $(x_1 + y_2) \cdot (x_1 + z_3) \cdot (y_3 + z_3)$ is not monotone either since it contains distinct literals y_2 and y_3 for variable Y. Complete reasons are known to be monotone and this property was exploited computationally in [4], [5] when computing the prime implicants and implicates of complete reasons.

D. Selection Semantics

For an instance $\mathcal{I}$ and its class formula Δ_i, we have [2]:

$$\mathcal{I} \models \forall \mathcal{I} \cdot \Delta_i \models \overline{\forall} \mathcal{I} \cdot \Delta_i \models \Delta_i$$

so the general reason is always weaker than the complete reason. The above relations also show that reasons, whether complete or general, are stronger than class formulas. Hence, the operators $\forall$ and $\overline{\forall}$ can be viewed as *selection operators* since they select subsets of the instances in class c_i [2], [7]. In particular, $\forall \mathcal{I} \cdot \Delta_i$ selects all instances in class c_i whose membership in the class does not depend on characteristics that are inconsistent with instance $\mathcal{I}$. For example, suppose that instance $\mathcal{I}'$ is in class c_i and disagrees with instance $\mathcal{I}$ only on the states of features X, Y, Z. Then $\mathcal{I}'$ will be selected by $\forall \mathcal{I} \cdot \Delta_i$ iff changing the states of any subset of its features X, Y, Z yields an instance that is also in class c_i. In contrast, $\overline{\forall} \mathcal{I} \cdot \Delta_i$ selects all instances that remain in class c_i if any of their characteristics are changed to agree with instance $\mathcal{I}$ (these include the ones selected by $\forall \mathcal{I} \cdot \Delta_i$). In the previous example, instance $\mathcal{I}'$ will be selected by $\overline{\forall} \mathcal{I} \cdot \Delta_i$ iff setting the states of any subset of its features X, Y, Z to the states they have in $\mathcal{I}$ yields an instance that is also in class c_i.

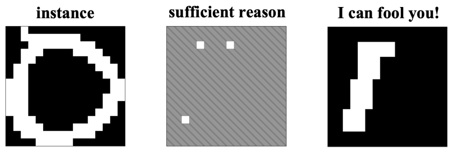


Fig. 4. Example sufficient reason from the domain of classifying digits [11].

IV. The Sufficient Reasons for a Decision

We now turn to answering question Q2: What minimal aspects of an instance guarantee the decision on that instance? We will provide two answers which depend on how we define "aspects." The first answer is based on the following notion.

Definition 9: Suppose instance $\mathcal{I}$ is in class c_i. The *sufficient reasons (SRs)* for the decision on instance $\mathcal{I}$ are the prime implicants of the complete reason $\forall \mathcal{I} \cdot \Delta_i$.

A sufficient reason is a term $\tau = \ell_1 \cdot \ldots \cdot \ell_n$ where each literal ℓ_i is simple (i.e., a state) and appears in instance $\mathcal{I}$.[5] This leads to the relationships $\mathcal{I} \models \tau \models \forall \mathcal{I} \cdot \Delta_i \models \Delta_i$ which further imply that any instance containing states $\ell_1, \ldots, \ell_n$ will be decided similarly to instance $\mathcal{I}$. Moreover, this does not hold for any strict subset of these states; otherwise, τ cannot be a prime implicant of $\forall \mathcal{I} \cdot \Delta_i$. In other words, the sufficient reason τ is a minimal set of characteristics in instance $\mathcal{I}$ that justify the decision on the instance, so other characteristics of the instance can be viewed as irrelevant to the decision as they can be changed in any fashion while sustaining the decision.

The notion of a sufficient reason was first introduced in [9] under the name of a *PI-explanation* using a different but equivalent definition. The term "sufficient reason" was first used in [5] which introduced the complete reason and showed that its prime implicants are the PI-explanations. Sufficient reasons are sometimes also called *abductive explanations* [10].

For a concrete example of sufficient reasons, from [11], consider Fig. 4 which depicts 16×16 images. The image on the left was passed to a Convolutional Neural Network (CNN) that is tasked with classifying digits $0, 1$ and the CNN classified the image correctly as digit 0. One of the sufficient reasons for this decision is depicted in the middle of Fig. 4. This sufficient reason contains 3 white pixels (out of 256 pixels) so any image that contains these 3 white pixels will be classified as digit 0 by this CNN, as shown on the right of Fig. 4.

For another concrete example, from [2], consider the classifier in Fig. 3 and the following patient, Lara,

$$\mathcal{I}\colon\ (\textsc{Diabetes}{=}\textsc{y}) \cdot (\textsc{Weight}{=}\textsc{over}) \cdot (\textsc{Btype}{=}\textsc{a}). \qquad (1)$$

The decision on Lara is $\textsc{yes}$ and the class formula $\Delta_{\textsc{yes}}$ is

$$\begin{aligned}&(\textsc{Diabetes}{=}\textsc{y})\cdot\\ &\quad[(\textsc{Weight}{=}\textsc{over}) +\\ &\quad(\textsc{Weight}{=}\textsc{norm}) \cdot (\textsc{Btype} \in \{\textsc{a}, \textsc{b}\}) +\\ &\quad(\textsc{Weight}{=}\textsc{under}) \cdot (\textsc{Btype} \in \{\textsc{a}, \textsc{b}, \textsc{ab}\})]. \qquad (2)\end{aligned}$$

[5]Since the complete reason is simple and fixated on the instance $\mathcal{I}$.

The complete reason $\forall\mathcal{I}\cdot\Delta_{\textsc{yes}}$ for this decision is

$$(\textsc{Diabetes}{=}\textsc{y})\cdot[(\textsc{Weight}{=}\textsc{over})+(\textsc{Btype}{=}\textsc{a})]. \quad (3)$$

It has two prime implicants

$$(\textsc{Diabetes}{=}\textsc{y})\cdot(\textsc{Weight}{=}\textsc{over}) \quad (4)$$
$$(\textsc{Diabetes}{=}\textsc{y})\cdot(\textsc{Btype}{=}\textsc{a}) \quad (5)$$

which are the sufficient reasons for the decision. Hence, the feature $\textsc{Btype}$ is irrelevant to the decision on Lara given her characteristics $(\textsc{Diabetes}{=}\textsc{y})\cdot(\textsc{Weight}{=}\textsc{over})$, and the feature $\textsc{Weight}$ is irrelevant given $(\textsc{Diabetes}{=}\textsc{y})\cdot(\textsc{Btype}{=}\textsc{a})$.

One can obtain weaker sufficient reasons—and, hence, identify further irrelevant aspects of instances—by employing the general reason behind the decision as shown recently in [2].

Definition 10: Suppose instance $\mathcal{I}$ is in class c_i. The *general sufficient reasons (GSRs)* for the decision on $\mathcal{I}$ are the variable-minimal prime implicants of the general reason $\overline{\forall}\mathcal{I}\cdot\Delta_i$.[6]

The general reason $\overline{\forall}\mathcal{I}\cdot\Delta_{\textsc{yes}}$ for the decision on Lara is

$$\begin{aligned}&(\textsc{Diabetes}{=}\textsc{y})\cdot\\ &\quad[(\textsc{Weight}{=}\textsc{over})+(\textsc{Btype}\in\{\textsc{a},\textsc{b}\})+\\ &\quad(\textsc{Weight}\in\{\textsc{under},\textsc{over}\})\cdot(\textsc{Btype}\in\{\textsc{a},\textsc{b},\textsc{ab}\})]\end{aligned} \quad (6)$$

and has three prime implicants

$$(\textsc{Diabetes}{=}\textsc{y})\cdot(\textsc{Weight}{=}\textsc{over}) \quad (7)$$
$$(\textsc{Diabetes}{=}\textsc{y})\cdot(\textsc{Btype}\in\{\textsc{a},\textsc{b}\}) \quad (8)$$
$$(\textsc{Diabetes}{=}\textsc{y})\cdot(\textsc{Weight}\in\{\textsc{under},\textsc{over}\})\cdot(\textsc{Btype}\in\{\textsc{a},\textsc{b},\textsc{ab}\}) \quad (9)$$

Only the first two are variable-minimal so they are the general sufficient reasons for the decision. We have seen (7) as a sufficient reason in (4). However, the general sufficient reason in (8) is weaker than the sufficient reason in (5) and conveys more information about the decision. In particular, the sufficient reason in (5) says that Lara's $\textsc{Weight}$ is irrelevant to the decision because she has $\textsc{Diabetes}$ and her $\textsc{Btype}$ type is $\textsc{a}$, but the general sufficient reason in (8) says it is because she has $\textsc{Diabetes}$ and her $\textsc{Btype}$ is not $\textsc{ab}$ or $\textsc{o}$.

Every general sufficient reason is consistent with the instance since the general reason is fixated on the instance. Define the *intersection* of instance $\mathcal{I}$ with a general sufficient reason τ as the smallest subset of $\mathcal{I}$ that implies τ. Then this intersection is guaranteed to be a sufficient reason. Moreover, this guarantee holds iff we have the variable-minimality condition in Def. 10. For example, the intersection of (1) and (8) is $(\textsc{Diabetes}{=}\textsc{y})\cdot(\textsc{Btype}{=}\textsc{a})$ which is a sufficient reason but the intersection of (1) and (9) is $(\textsc{Diabetes}{=}\textsc{y})\cdot(\textsc{Weight}{=}\textsc{over})\cdot(\textsc{Btype}{=}\textsc{a})$ which is not a sufficient reason.[7] More generally, the sufficient reasons of a decision are precisely the intersections of general sufficient reasons with the instance. Hence, general sufficient reasons subsume their classical counterparts and provide more information about decisions and underlying classifiers. We finally note, again, that if all features are binary, the complete and general reasons are equivalent so general sufficient reasons reduce to classical sufficient reasons. All the previous observations are implied by the results in [2].

[6]This definition is a proposition in [2] which provided a different definition for general sufficient reasons that does not reference general reasons.

[7]Ref. [3] suggested using prime implicants that are not variable-minimal.

V. The Necessary Reasons for a Decision

We now turn to answering question Q3: What minimal changes to an instance will lead to a different decision on that instance? The answer to this question will identify minimal sets of features that will flip the decision if changed properly. There are two orthogonal variations on this question. The first determines whether the specific changes to these features are also identified which in turn depends on whether we employ the complete or general reason for this purpose. The second variation relates to whether we seek to flip the decision into some specific alternative or some arbitrarily different decision.

A. Undermining a Current Decision

We assume in this section that our goal is to undermine the current decision, without targeting a particular alternate decision. This aim can be achieved using the following notion.

Definition 11: Suppose instance $\mathcal{I}$ is in class c_i. The *necessary reasons (NRs)* for the decision on instance $\mathcal{I}$ are the prime implicates of the complete reason $\forall\mathcal{I}\cdot\Delta_i$.

Each necessary reason is a clause $\ell_1+\ldots+\ell_n$ where each literal ℓ_i is simple (a state) and appears in instance $\mathcal{I}$.[8] Hence, each necessary reason is satisfied by the instance. However, if we change the instance in a way that violates a necessary reason, the decision is no longer guaranteed to be sustained. Suppose state ℓ_i is for feature X_i. To violate the necessary reason $\ell_1+\ldots+\ell_n$ we must change the state of each feature X_i in the instance. The guarantee that comes with a necessary reason is that at least one such change will flip the decision, and no strict subset of this change will flip the decision.

The notion of a necessary reason was first formalized in [12] using a different but equivalent definition, under the name of a *contrastive explanation* [13]. Necessary reasons are sometimes also called *counterfactual explanations* [14].[9] The term "necessary reason" was first used in [4] which showed that contrastive explanations are the prime implicates of the complete reason and provided the semantics of necessity discussed earlier (a property that must be preserved if the decision is to be preserved). Let us now illustrate necessary reasons using a concrete example.

We are back to patient Lara given in (1) who was decided as $\textsc{yes}$ by the classifier in Fig. 3. The class formula is given in (2) and the complete reason for the decision is given in (3). This complete reason has two prime implicates, $(\textsc{Diabetes}{=}\textsc{y})$ and $(\textsc{Weight}{=}\textsc{over})+(\textsc{Btype}{=}\textsc{a})$. There is only one way to violate the first reason by setting $\textsc{Diabetes}$ to $\textsc{n}$, which will indeed flip the decision to $\textsc{no}$ as can be verified in Fig. 3. The

[8]Since the complete reason is simple and fixated on the instance.

[9]There is an extensive body of work in philosophy, social science and AI that discusses contrastive explanations and counterfactual explanations; see, e.g., [15]–[24]. While the definitions of these notions are sometimes variations or refinements on one another, they are not always compatible.

second necessary reason can be violated in six different ways, by setting $\textsc{Weight}$ to a state in $\{\textsc{under}, \textsc{norm}\}$ and setting $\textsc{Btype}$ to a state in $\{\textsc{b}, \textsc{ab}, \textsc{o}\}$. Not all of these changes/violations will flip the decision but at least one will. The change to $(\textsc{Weight}{=}\textsc{under}) \cdot (\textsc{Btype}{=}\textsc{o})$ does flip the decision but the change to $(\textsc{Weight}{=}\textsc{under}) \cdot (\textsc{Btype}{=}\textsc{ab})$ does not (always assuming that $\textsc{Diabetes}$ is left unchanged). We stress again that no strict subset of the first change will flip the decision since the changes suggested by necessary reasons are minimal.

We will now see that we can do better than this if we employ the general reasons of decisions as suggested recently in [2].

Definition 12: Suppose instance $\mathcal{I}$ is in class c_i. The *general necessary reasons (GNRs)* for the decision on $\mathcal{I}$ are the variable-minimal prime implicates of general reason $\overline{\forall}\mathcal{I}\cdot\Delta_i$.[10]

Like necessary reasons, violating a general necessary reason will undermine the decision. However, we now have an additional guarantee: *every* violation of a general necessary reason will flip the decision. This guarantee will not hold if we do not insist on variable-minimal prime implicates, and if some prime implicate that is not variable minimal satisfies this guarantee, then some changes it suggests cannot be minimal (i.e., we can flip the decision with fewer changes to the instance). Moreover, any change that flips the decision and is suggested by some necessary reason will also be suggested by some general necessary reason. That is, the latter reasons subsume the former and convey more information about decisions and the underlying classifiers.[11] Let us look at a concrete example.

The general reason for the decision on Lara is given in (6) and has three prime implicates

$$(\textsc{Diabetes}{=}\textsc{y})$$
$$(\textsc{Weight}{=}\textsc{over}) + (\textsc{Btype} \in \{\textsc{a}, \textsc{b}, \textsc{ab}\})$$
$$(\textsc{Weight} \in \{\textsc{under}, \textsc{over}\}) + (\textsc{Btype} \in \{\textsc{a}, \textsc{b}\}).$$

All are variable-minimal so they are all general necessary reasons. The first can be violated in only one way by setting $\textsc{Diabetes}$ to $\textsc{n}$ which will flip the decision. The second can be violated in two ways by setting $\textsc{Weight}$ to a state in $\{\textsc{under}, \textsc{norm}\}$ and setting $\textsc{Btype}$ to state $\textsc{o}$. Both violations will flip the decision as can be verified using the classifier in Fig. 3. The third general necessary reason can be violated in two ways by setting $\textsc{Weight}$ to state $\textsc{norm}$ and setting $\textsc{Btype}$ to a state in $\{\textsc{ab}, \textsc{o}\}$. Again, both violations will flip the decision. Hence, every violation of these general necessary reasons is guaranteed to flip the decision. Moreover, every change suggested by a necessary reason and which flips the decision is also suggested by some general necessary reason.

We conclude this section by the following remarks. Boolean resolution derives the clause $\alpha{+}\beta$ from clauses $x{+}\alpha$ and $\overline{x}{+}\beta$. Moreover, a classical method for computing prime implicates of Boolean CNFs is to close the CNF under resolution and remove subsumed clauses after each resolution step; see, e.g., [25], [26]. As recently shown in [2], this can also be done for discrete CNFs but using a generalized resolution rule that derives clause $\ell{\cdot}\ell'{+}\alpha{+}\beta$ from clauses $\ell{+}\alpha$ and $\ell'{+}\beta$ where ℓ and ℓ' are literals for the same variable. What is particularly striking is this. If the CNF is locally fixated (Def. 7), as is the case for general reasons, then one can compute the general necessary reasons by simply discarding clauses that are not variable-minimal after each resolution step [2].

B. Targeting a New Decision

The previous discussion showed how (general) necessary reasons can be used to flip a decision but without a guarantee on what the new decision is. This becomes an issue when the classifier has more than two classes and hence can make more than two decisions. Consider a classifier that decides whether an applicant should be approved for large loan (c_1), approved for a small loan (c_2) or declined (c_3). Let $\Delta_1, \Delta_2, \Delta_3$ be the corresponding class formulas and suppose we have an applicant $\mathcal{I}$ who was approved for a small loan, that is, $\mathcal{I} \models \Delta_2$. The (general) necessary reasons for this decision do suggest minimal changes to the application that will flip the decision but they do not guarantee whether the new decision will approve a larger loan or decline the application. Suppose that we wish to flip the decision so the applicant is approved for a larger loan (c_1). What we can do is merge classes c_2 and c_3 leading to a classifier with two decisions c_1, c_{23} and corresponding class formulas Δ_1 and $\Delta_{23} = \Delta_2 + \Delta_3$. The decision to approve a small loan (c_2) in the original classifier, $\mathcal{I} \models \Delta_2$, is now a decision to approve a small loan or decline the application (c_{23}) in the new classifier, $\mathcal{I} \models \Delta_{23}$. Flipping this decision is then guaranteed to approve a large loan (c_1).

More generally, suppose we have a classifier with classes $c_1, \ldots, c_n$, class formulas $\Delta_1, \ldots, \Delta_n$, and an instance $\mathcal{I}$ in class c_i, $\mathcal{I} \models \Delta_i$. If we wish to minimally change this instance so it moves to some other class c_j, we need to compute the complete reason $\forall\mathcal{I} \cdot \sum_{k \neq j} \Delta_k$ or the general reason $\overline{\forall}\mathcal{I}\cdot\sum_{k \neq j} \Delta_k$. This can then be followed by computing the (variable-minimal) prime implicates of these reasons as discussed earlier. This technique of merging class formulas was proposed in [4] and the resulting necessary reasons have been known as *targeted contrastive explanations* [12].

VI. Compiling Class Formulas

A comprehensive discussion of compiling class formulas for various types of classifiers is beyond the scope of this tutorial.[12] However, we will share some key insights about this compilation process to make it somewhat less mysterious. The first observation is that the techniques underlying this process depend on the type of classifier, and its difficulty depends on both the type of classifier and the desired form of compiled formulas. A number of works have targeted compilations in the form of *tractable circuits* [27] but we will largely ignore this dimension here since compiling formulas into tractable circuits is a well understood problem that has been studied extensively in the area of *knowledge compilation* [28].

[10]This definition is a proposition in [2] which provided a different definition for general necessary reasons that does not reference general reasons.

[11]The general necessary reasons are fixated (Def. 7) on instance $\mathcal{I}$.

[12]The term “class formulas” was first used in [4]. Compiling the “decision function” or “input-output behavior” of a classifier are more common terms.

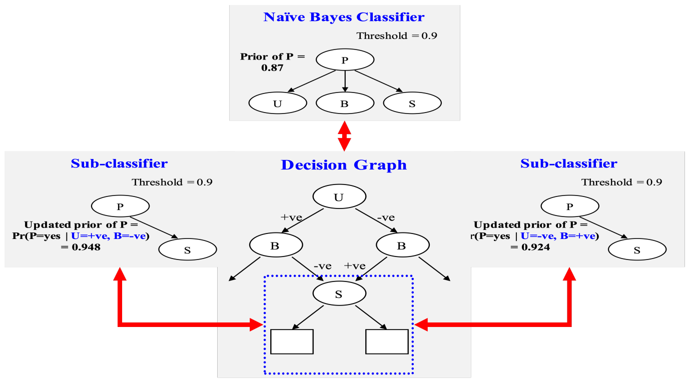


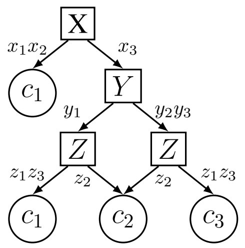
Fig. 5. Compiling a Naïve Bayes classifier into a decision graph [29].

A. Bayesian Networks

Compiling the class formulas of Bayesian network classifiers is perhaps the most subtle conceptually since deciding what instance belongs to what class requires probabilistic reasoning. There are two fundamental insights behind this compilation process. The first is the notion of *equivalence intervals* introduced in [29]. Consider the Naïve Bayes classifier in Fig. 1(a) which has the class distribution $\Pr(P{=}\text{YES}), \Pr(P{=}\text{NO}) = (0.87, 0.13)$. We can change this distribution quite significantly without changing any decision made by this classifier as long as $\Pr(P{=}\text{YES}) \in [0.684, 0970]$, which is called an equivalence interval. The second fundamental insight is the notion of *equivalent sub-classifiers* also introduced in [29]. Continuing with the same classifier, suppose we set the features U, B to values +VE, −VE. This leads to a sub-classifier over one features S with a new class distribution $(0.948, 0.052)$ since $\Pr(P{=}\text{YES}|U{=}{+}\text{VE}, B{=}{-}\text{VE}) = 0.948$. Similarly, if these two features are set to −VE, +VE, we get another sub-classifier over the same features U but with a different class distribution $(0.924, 0.076)$ since $\Pr(P{=}\text{YES}|U{=}{-}\text{VE}, B{=}{+}\text{VE}) = 0.924$. By utilizing the concept of equivalence intervals, we can conclude that these two sub-classifiers are equivalent, that is, they make the same decisions on sub-instances $S{=}{+}\text{VE}$ and $S{=}{-}\text{VE}$. Putting these two insights together, we can compile a Bayesian network classifier into a (symbolic) decision graph by conducting a depth-first search on the space of feature instantiations while pruning the search whenever we encounter a sub-classifier that is equivalent to another sub-classifier that we already encountered (compiled); see Fig. 5. This was used to compile Naïve Bayes classifiers in [29], tree-structured classifiers in [9] and graph-structured classifiers in [30]. The techniques for computing equivalence intervals and for identifying equivalent sub-classifiers depend on the classifier's structure which explains this progression. Fig. 1(b) depicts the compiled decision graph using this method for the Naïve Bayes classifier in Fig. 1(a). Extracting class formulas from decision graphs is discussed next.

B. Decision Graphs and Random Forests

Our next discussion on decision graphs applies to decision trees since they are a special case. It is well known that the class formulas of decision graphs can be directly obtained as DNFs. To construct the DNF for a class c_i, we simply construct a term for each path from the root to a leaf labeled with c_i and then disjoin these terms. For example, in the decision graph below, there are two paths from the root to class c_1 which generate the DNF $\Delta_1 = x_{12} + x_3 \cdot y_1 \cdot z_{13}$. For decision graphs (that are not trees), this may yield a DNF that is exponentially larger than the graph. We can alleviate this by constructing an NNF circuit whose size is guaranteed to be linear in the decision graph size. The next construction from [4] produces circuits that satisfy even stronger properties as such circuits will allow one, under weak conditions, to compute complete and general reasons in time linear in the circuit size.



Let $c_1, \ldots, c_n$ be the classes of the decision graph (i.e., labels of leaf nodes) and suppose we wish to construct an NNF circuit for the formula of some class c_k. We first define the function $\mathtt{nnf}(N)$ which maps a node N in the decision graph into an NNF fragment as follows. If node N has outgoing edges $\xrightarrow{\ell_1} C_1, \ldots, \xrightarrow{\ell_m} C_m$, then $\mathtt{nnf}(N) = \sum_{i=1}^{m}(\ell_i \cdot \mathtt{nnf}(C_i))$. For leaf nodes, $\mathtt{nnf}(c_i) = \top$ if $c_i = c_k$ and $\mathtt{nnf}(c_i) = \bot$ if $c_i \neq c_k$. We can now convert the decision graph into an NNF circuit by calling $\mathtt{nnf}(R)$ where R is the graph's root node. This is the standard method but [4] defined the function $\mathtt{nnf}(.)$ differently so the NNF circuits satisfy desirable properties. In particular, for an internal node N, it instead used $\mathtt{nnf}(N) = \prod_{i=1}^{m}(\ell'_i + \mathtt{nnf}(C_i))$ where literal ℓ'_i is the complement of literal ℓ_i. Leaf nodes are kept the same, $\mathtt{nnf}(c_i) = \top$ if $c_i = c_k$ and $\mathtt{nnf}(c_i) = \bot$ if $c_i \neq c_k$. The new method is equivalent to using the first method to construct an NNF circuit for the union of classes $c_1, \ldots, c_{k-1}, c_{k+1}, \ldots, c_n$ and then negating the resulting circuit using deMorgan's law.

The NNF circuits constructed by this new method are guaranteed to be or-decomposable (Def. 4) if the decision graph satisfies the *test-once property* (a feature is tested at most once on any path). Recall that complete and general reasons can be computed in linear time if the class formulas are or-decomposable NNFs since the operators $\forall$ (Def. 3) and $\overline{\forall}$ (Def. 6) will distribute over disjunctions and conjunctions in such NNFs. One can also obtain NNF circuits that allow linear-time computation of complete and general reasons if the decision graph satisfies the *weak test-once property* discussed in [4]. Discretized decision graphs satisfy this property (e.g., the decision tree in Fig. 2(b)). The above construction and its variants are the basis for the closed-form complete reasons proposed in [4] and the closed-form general reasons in [2].

For random forests with majority voting, one can easily construct NNF circuits for class formulas by combining the NNF circuits for trees in the forest using a majority circuit. But the resulting circuit is not guaranteed to be or-decomposable even when the circuits for trees are or-decomposable.

C. Neural Networks

The compilation of neural network classifiers into class formulas is more involved due to the multiplicity of techniques and assumptions such as the type of activation functions and whether the network is binary, binarized or quantized. We will therefore restrict the discussion to one approach for binary neural networks while giving pointers to other approaches.

The work we shall sample is [11] which assumed neural networks with binary inputs and step-activation functions. The compilation technique is based on a few observations. First, a neuron with step activation has a binary output so if its inputs are also binary then the neuron represents a Boolean function. Hence, a neural network with binary inputs and step-activation functions must represent a Boolean function (i.e., the signals on its inputs, internal wires and output must all be in $\{0, 1\}$). One can therefore convert the neural network into a Boolean circuit (from which class formulas can be easily extracted) if one can compile a neuron into a Boolean circuit (i.e., one with binary inputs and a step-activation function). The next observation is that a neuron with step activation is a linear classifier similar to Naïve Bayes classifiers. Hence, the technique we discussed earlier for compiling Naïve Bayes classifiers into decision graphs, from [29], can be adopted for compiling this class of neurons into decision graphs and then NNF circuits. Once such neurons are successfully compiled, the neural network can be immediately represented as a Boolean circuit from which class formulas can be easily obtained. This work went a bit further by employing a variant on the compilation method in [29] which assumes that the neuron weights $w_1, \ldots, w_n$ and threshold T are integers. This allows one to conduct the compilation in $O(nW)$ time where $W = |T| + \sum_{i=1}^{n} |w_i|$ is a sum of absolute values. This pseudo-polynomial time compilation algorithm can be applied to real-valued weights by multiplying the weights by a constant and then truncating (i.e., the weights have fixed precision). This technique permitted the compilation of neural networks with hundreds of inputs (features). The example in Fig. 4 and many other examples in [11] were produced by this approach.

For further techniques that encode input-output behavior symbolically, see [31]–[37] for binarized neural networks and [38]–[41] for quantized ones. Most of these works target verification tasks though instead of explaining behavior.

VII. Concluding Remarks

We close this tutorial with three remarks. The first remark concerns two distinct classes of works on explainable AI. One class assumes enough information about the classifier to allow the construction of class formulas or, more generally, the characterization of which instances belong to what classes. This is clearly the assumption we made in this tutorial and the resulting approaches are known as *model-based.* These approaches usually seek explanations that come with hard guarantees like the ones we discussed. The other class of works assume that we can only query the classifier, that is, ask it to classify instances. These approaches are known as *model-agnostic* and have been mostly popularized through the early systems described in [42], [43]. These approaches tend to scale better but do not offer hard guarantees and can be viewed as computing approximate explanations [44].

The second remark relates to the extensive nature of investigations that were conducted on explainability over the last few years, particularly on the complexity of computing explanations. The following are some examples. For Naïve Bayes (and linear) classifiers, it was shown that one sufficient reason can be generated in log-linear time, and all sufficient reasons can be generated with polynomial delay [45]. For decision trees, the complexity of generating one sufficient reason was shown to be in polynomial time [46]. Later works showed the same complexity for decision graphs [47] and some classes of tractable circuits [14], [48]. The generation of sufficient reasons for decision trees was also studied in [49], including the generation of shortest sufficient reasons which was shown to be hard even for a single reason. The generation of shortest sufficient reasons was also studied in a broader context that includes decision graphs and SDDs [4].[13] The complexity of shortest sufficient reasons was studied in [55] for Boolean classifiers which correspond to decision graphs and for neural networks with ReLU activation functions. It was further shown that the number of necessary reasons is linear in the decision tree size [4], [47], that all such reasons can be computed in polynomial time [4], [49], and that the shortest necessary reasons can be enumerated with polynomial delay if the classifier satisfies some conditions as stated in [56]. Further complexity results were shown in [14], [48], where classifiers where categorized based on the tractable circuits that represent them [48] or the kinds of processing they permit in polynomial time [14]. A comprehensive study of complexity was also presented in [57] for a large set of explanation queries and classes of Boolean classifiers. Computational approaches based on either SAT, MaxSAT or partial MaxSAT were also proposed for random forests, e.g., [58], [59], tree ensembles, e.g., [60] and boosted trees, e.g., [61]. Formal results on explainability were even employed to question common wisdoms like those relating to the interpretability of decision trees [62].

The last remark relates to the storyline adopted in this tutorial which treated the complete and general reasons behind decisions as the core notions in this theory of explainability and used them to describe other notions, even ones that were proposed before such reasons were conceived. This is an outcome of our firm belief that the notion of "instance abstraction" must be the core of any comprehensive and well-founded theory of explainability. We hope the reader would agree with us that this treatment has also led to a minimalistic formulation that explicates semantics and facilitates computation.

Acknowledgment

I wish to thank Yizuo Chen, Haiying Huang and Albert Ji for their useful feedback and discussions.

[13]SDDs (*Sentential Decision Diagrams*) are decision diagrams that branch on formulas (sentences) instead of variables [50]. SDDs are a superset of, and exponentially more succinct than [51], OBDDs but are not comparable to some other types of decision graphs in terms of succinctness [52]–[54].

References

[1] H. Chan and A. Darwiche, "Reasoning about Bayesian network classifiers," in *UAI*. Morgan Kaufmann, 2003, pp. 107–115.

[2] C. Ji and A. Darwiche, "A new class of explanations for classifiers with non-binary variables," *CoRR*, vol. abs/2304.14760, 2020.

[3] A. Choi, A. Shih, A. Goyanka, and A. Darwiche, "On symbolically encoding the behavior of random forests," *CoRR*, vol. abs/2007.01493, 2020.

[4] A. Darwiche and C. Ji, "On the computation of necessary and sufficient explanations," in *AAAI*. AAAI Press, 2022.

[5] A. Darwiche and A. Hirth, "On the reasons behind decisions," in *ECAI*, ser. Frontiers in Artificial Intelligence and Applications, vol. 325. IOS Press, 2020, pp. 712–720.

[6] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Computers*, vol. 35, no. 8, pp. 677–691, 1986.

[7] A. Darwiche and P. Marquis, "On quantifying literals in Boolean logic and its applications to explainable AI," *J. Artif. Intell. Res.*, vol. 72, pp. 285–328, 2021.

[8] A. Darwiche, "Decomposable negation normal form," *J. ACM*, vol. 48, no. 4, pp. 608–647, 2001.

[9] A. Shih, A. Choi, and A. Darwiche, "A symbolic approach to explaining bayesian network classifiers," in *IJCAI*. ijcai.org, 2018, pp. 5103–5111.

[10] A. Ignatiev, N. Narodytska, and J. Marques-Silva, "Abduction-based explanations for machine learning models," in *Proceedings of the Thirty-Third Conference on Artificial Intelligence (AAAI)*, 2019, pp. 1511–1519.

[11] W. Shi, A. Shih, A. Darwiche, and A. Choi, "On tractable representations of binary neural networks," in *KR*, 2020, pp. 882–892.

[12] A. Ignatiev, N. Narodytska, N. Asher, and J. Marques-Silva, "From contrastive to abductive explanations and back again," in *AI*IA*, ser. Lecture Notes in Computer Science, vol. 12414. Springer, 2020, pp. 335–355.

[13] P. Lipton, "Contrastive explanation," *Royal Institute of Philosophy Supplements*, vol. 27, p. 247–266, 1990.

[14] G. Audemard, F. Koriche, and P. Marquis, "On tractable XAI queries based on compiled representations," in *KR*, 2020, pp. 838–849.

[15] A. Garfinkel, "Forms of explanation: Rethinking the questions in social theory," *British Journal for the Philosophy of Science*, vol. 33, no. 4, pp. 438–441, 1982.

[16] D. Lewis, "Causal explanation," in *Philosophical Papers Vol. Ii*, D. Lewis, Ed. Oxford University Press, 1986, pp. 214–240.

[17] D. Temple, "The contrast theory of why-questions," *Philosophy of Science*, vol. 55, no. 1, pp. 141–151, 1988.

[18] S. Wachter, B. D. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *Harvard Journal of Law & Technology*, vol. 31, no. 2, 2018.

[19] J. van der Waa, M. Robeer, J. van Diggelen, M. Brinkhuis, and M. Neerincx, "Contrastive explanations with local foil trees," *arXiv preprint arXiv:1806.07470*, 2018.

[20] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, 2019.

[21] B. Mittelstadt, C. Russell, and S. Wachter, "Explaining explanations in AI," *Proceedings of the Conference on Fairness, Accountability, and Transparency*, Jan 2019.

[22] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee, "Counterfactual visual explanations," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 2376–2384.

[23] S. Verma, V. Boonsanong, M. Hoang, K. E. Hines, J. P. Dickerson, and C. Shah, "Counterfactual explanations and algorithmic recourses for machine learning: A review," 2022.

[24] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, Jan 2020.

[25] V. Gurvich and L. Khachiyan, "On generating the irredundant conjunctive and disjunctive normal forms of monotone Boolean functions," *Discrete Applied Mathematics*, vol. 96, pp. 363–373, 1999.

[26] Y. Crama and P. L. Hammer, "Boolean functions - theory, algorithms, and applications," in *Encyclopedia of mathematics and its applications*, 2011.

[27] A. Darwiche, "Three modern roles for logic in AI," in *PODS*. ACM, 2020, pp. 229–243.

[28] A. Darwiche and P. Marquis, "A knowledge compilation map," *J. Artif. Intell. Res.*, vol. 17, pp. 229–264, 2002.

[29] H. Chan and A. Darwiche, "Reasoning about bayesian network classifiers," in *UAI*. Morgan Kaufmann, 2003, pp. 107–115.

[30] A. Shih, A. Choi, and A. Darwiche, "Compiling bayesian network classifiers into decision graphs," in *AAAI*. AAAI Press, 2019, pp. 7966–7974.

[31] N. Narodytska, S. P. Kasiviswanathan, L. Ryzhyk, M. Sagiv, and T. Walsh, "Verifying properties of binarized deep neural networks," in *AAAI*. AAAI Press, 2018, pp. 6615–6624.

[32] N. Narodytska, H. Zhang, A. Gupta, and T. Walsh, "In search for a sat-friendly binarized neural network architecture," in *ICLR*. OpenReview.net, 2020.

[33] N. Narodytska, A. A. Shrotri, K. S. Meel, A. Ignatiev, and J. Marques-Silva, "Assessing heuristic machine learning explanations with model counting," in *SAT*, ser. Lecture Notes in Computer Science, vol. 11628. Springer, 2019, pp. 267–278.

[34] T. Baluta, S. Shen, S. Shinde, K. S. Meel, and P. Saxena, "Quantitative verification of neural networks and its security applications," in *CCS*. ACM, 2019, pp. 1249–1264.

[35] A. Shih, A. Darwiche, and A. Choi, "Verifying binarized neural networks by angluin-style learning," in *SAT*, ser. Lecture Notes in Computer Science, vol. 11628. Springer, 2019, pp. 354–370.

[36] K. Jia and M. Rinard, "Efficient exact verification of binarized neural networks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[37] Y. Zhang, Z. Zhao, G. Chen, F. Song, and T. Chen, "BDD4BNN: A BDD-based quantitative analysis framework for binarized neural networks," in *Computer Aided Verification*, A. Silva and K. R. M. Leino, Eds. Cham: Springer International Publishing, 2021, pp. 175–200.

[38] M. Giacobbe, T. A. Henzinger, and M. Lechner, "How many bits does it take to quantize your neural network?" in *Tools and Algorithms for the Construction and Analysis of Systems*, A. Biere and D. Parker, Eds. Cham: Springer International Publishing, 2020, pp. 79–97.

[39] T. A. Henzinger, M. Lechner, and D. Zikelic, "Scalable verification of quantized neural networks," in *AAAI*. AAAI Press, 2021, pp. 3787–3795.

[40] Y. Zhang, Z. Zhao, G. Chen, F. Song, M. Zhang, T. Chen, and J. Sun, "QVIP: An ilp-based formal verification approach for quantized neural networks," ser. ASE '22. New York, NY, USA: Association for Computing Machinery, 2023.

[41] Y. Zhang, F. Song, and J. Sun, "QEBVerif: Quantization error bound verification of neural networks," *CoRR*, vol. abs/2212.02781, 2022.

[42] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," in *KDD*. ACM, 2016, pp. 1135–1144.

[43] ——, "Anchors: High-precision model-agnostic explanations," in *AAAI*. AAAI Press, 2018, pp. 1527–1535.

[44] A. Ignatiev, N. Narodytska, and J. Marques-Silva, "On validating, repairing and refining heuristic ML explanations," *CoRR*, vol. abs/1907.02509, 2019.

[45] J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, and N. Narodytska, "Explaining naive bayes and other linear classifiers with polynomial time and delay," in *NeurIPS*, 2020.

[46] Y. Izza, A. Ignatiev, and J. Marques-Silva, "On explaining decision trees," *CoRR*, vol. abs/2010.11034, 2020.

[47] X. Huang, Y. Izza, A. Ignatiev, and J. Marques-Silva, "On efficiently explaining graph-based classifiers," in *KR*, 2021, pp. 356–367.

[48] X. Huang, Y. Izza, A. Ignatiev, M. C. Cooper, N. Asher, and J. Marques-Silva, "Efficient explanations for knowledge compilation languages," *CoRR*, vol. abs/2107.01654, 2021.

[49] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Lagniez, and P. Marquis, "On the explanatory power of boolean decision trees," *Data & Knowledge Engineering*, vol. 142, p. 102088, 2022.

[50] A. Darwiche, "SDD: A new canonical representation of propositional knowledge bases," in *IJCAI*. IJCAI/AAAI, 2011, pp. 819–826.

[51] S. Bova, "SDDs are exponentially more succinct than obdds," in *AAAI*. AAAI Press, 2016, pp. 929–935.

[52] B. Bollig and M. Buttkus, "On the relative succinctness of sentential decision diagrams," *Theory Comput. Syst.*, vol. 63, no. 6, pp. 1250–1277, 2019.

[53] P. Beame and V. Liew, "New limits for knowledge compilation and applications to exact model counting," in *UAI*. AUAI Press, 2015, pp. 131–140.

[54] P. Beame, J. Li, S. Roy, and D. Suciu, "Lower bounds for exact model counting and applications in probabilistic databases," in *UAI*. AUAI Press, 2013.

[55] P. Barceló, M. Monet, J. Pérez, and B. Subercaseaux, "Model interpretability through the lens of computational complexity," in *NeurIPS*, 2020.

[56] G. Audemard, F. Koriche, and P. Marquis, "On Tractable XAI Queries based on Compiled Representations," in *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, 9 2020, pp. 838–849.

[57] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J. Lagniez, and P. Marquis, "On the computational intelligibility of Boolean classifiers," in *KR*, 2021, pp. 74–86.

[58] Y. Izza and J. Marques-Silva, "On explaining random forests with SAT," in *IJCAI*. ijcai.org, 2021, pp. 2584–2591.

[59] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J. Lagniez, and P. Marquis, "Trading complexity for sparsity in random forest explanations," in *AAAI*. AAAI Press, 2022, pp. 5461–5469.

[60] A. Ignatiev, Y. Izza, P. J. Stuckey, and J. Marques-Silva, "Using maxsat for efficient explanations of tree ensembles," in *AAAI*. AAAI Press, 2022, pp. 3776–3785.

[61] G. Audemard, J. Lagniez, P. Marquis, and N. Szczepanski, "Computing abductive explanations for boosted trees," ser. Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS), vol. 206. PMLR, 2023.

[62] Y. Izza, A. Ignatiev, and J. Marques-Silva, "On tackling explanation redundancy in decision trees," *J. Artif. Intell. Res.*, vol. 75, pp. 261–321, 2022.