On the Computational Intelligibility of Boolean Classifiers

Gilles Audemard¹, Steve Bellart¹, Louenas Bounia¹, Frédéric Koriche¹, Jean-Marie Lagniez¹, Pierre Marquis¹,²

¹CRIL, Université d’Artois & CNRS, France
²Institut Universitaire de France
{audemand, bellart, bounia, koriche, lagniez, marquis}@cril.fr

Abstract

In this paper, we investigate the computational intelligibility of Boolean classifiers, characterized by their ability to answer XAI queries in polynomial time. The classifiers under consideration are decision trees, DNF formulae, decision lists, decision rules, tree ensembles, and Boolean neural nets. Using 9 XAI queries, including both explanation queries and verification queries, we show the existence of a large intelligibility gap between the families of classifiers. On the one hand, all the 9 XAI queries are tractable for decision trees. On the other hand, none of them is tractable for DNF formulae, decision lists, random forests, boosted decision trees, Boolean multilayer perceptrons, and binarized neural networks.

1 Introduction

What is a good classifier? Such a common question calls for the identification of several criteria, in order to assess the quality of classifiers. To this point, a key criterion for measuring the generalization ability of classifiers is accuracy. To this point, a key criterion for measuring the generalization ability of classifiers is accuracy. Given a probability distribution over data instances, the accuracy of a (Boolean) classifier is defined by the probability of correctly labeling a random data instance. In statistical learning, the underlying probability distribution is unknown, and we only have access to a data sample for training the classifier. The learning problem is thus cast as a stochastic optimization task: given a family of candidate classifiers, often referred to as the concept class, find a classifier in the family that minimizes the (possibly regularized) empirical error on the training sample. In practice, the classification accuracy is estimated on test samples using evaluation metrics such as, for example, stratified cross-validation.

However, accuracy is not the sole criterion for choosing a classifier: in many real-world applications, another important criterion is intelligibility. Roughly speaking, a classifier is intelligible if its predictions can be explained in understandable terms to a user, and if its behavior can be verified according to the user’s expectations. The explainability requirement is a legal issue in Europe since the implementation of the General Data Protection Regulation (EU) 2016/679 (“GDPR”) on May 25th, 2018 (Goodman and Flaxman 2017). Accordingly, explainable and robust AI (XAI) has become a very active research topic for the past couple of years (see e.g. (Bunel et al. 2018; Shih, Choi, and Darwiche 2018; Plumb, Molitor, and Talwalkar 2018; Ignatiev, Narodytska, and Marques-Silva 2019; Chen et al. 2019; Srinivasan, Vig, and Bain 2019; Shih, Darwiche, and Choi 2019; Crabbe et al. 2020; Horel and Giesecke 2020; Jia and Rinard 2020; Marques-Silva et al. 2020; Ramamurthy et al. 2020)).

Despite the importance of both criteria, intelligibility appears to be much more difficult to circumscribe than accuracy. Indeed, in the statistical learning literature, the generalization ability of classifiers has been formally characterized through the prisms of learnability (Valiant 1984; Haussler 1992), uniform convergence (Vapnik 1998), and algorithmic stability (Shalev-Shwartz et al. 2010; Charles and Papailiopoulos 2018). By contrast, the term “intelligibility” holds no agreed upon meaning, since it depends on various desiderata for clarifying the classifier behavior in some practical situations (Lipton 2018). Yet, different forms of intelligibility can be formalized, by focusing on the classifier ability to properly answer questions. Such forms of intelligibility are reflected by explanation and verification queries introduced so far in the XAI literature. Notably, a classifier should be equipped with explanation facilities including, for example, the ability to identify few relevant features which together are sufficient for predicting the label of a data instance. Furthermore, the classifier should be amenable to inspection, especially when the user has some expectations about the behavior of the classifier, and she is interested in checking whether the classifier complies to those expectations. For instance, in a loan classification problem, if a loan is granted to an applicant who does not have a high income, then loan should not be denied when the income increases, provided that the other features are unchanged. This expectation can be formalized using a verification query that checks the monotonic behavior of the classifier on the feature related to the applicant’s income.

Addressing such XAI queries requires the availability of inference algorithms for computing answers in reasonable time. From this perspective, each query can be viewed as a property that a family of classifiers may offer or not: it is offered when there exists a polynomial-time algorithm to answer the query from any classifier of the family, and it is not when there is no such algorithm, unless \(P = NP\). In other words, the computational intelligibility of a family of classifiers can be defined as the set of tractable XAI queries supported by the family, leading to an intelligibil-
ity map when several families of classifiers are considered. Such an approach echoes the computational evaluation of KR languages achieved in the knowledge compilation map (Darwiche and Marquis 2002).

The aim of this paper is to pave the way for the computational intelligibility of Boolean classifiers. As a baseline, we use 9 XAI queries from those considered in (Audemard, Koriche, and Marquis 2020), which are summarized in Table 1: EMC, DPI, and ECO are explanation queries, and CIN, EIN, IMA, IIR, IMO, MCP are verification queries. Based on this portfolio of XAI queries, we examine 7 families of Boolean classifiers: decision trees, DNF formulae, decision lists, random forests, boosted decision trees, Boolean multilayer perceptrons, and Boolean neural networks. The main contribution of this paper lies in a number of complexifications and any subset where $\mathcal{B}^n \ni x$.

For a positive integer $n$, let $[n]$ denote the set $\{1, \cdots , n\}$. Let $\mathcal{F}_n$ be the set of all Boolean functions from $\mathcal{B}^n$ into $\mathcal{B}$, where $\mathcal{B} = \{0, 1\}$. Any member $f$ of $\mathcal{F}_n$ is called a concept, and any subset $\mathcal{F}$ of $\mathcal{F}_n$ is called a concept class. Any vector $x$ in the Boolean hypercube $\mathcal{B}^n$ is called an instance; $x$ is a positive example (or model) of some concept $f$ if $f(x) = 1$, and $x$ is a negative example of $f$ if $f(x) = 0$. In what follows, we use $\top$ and $\bot$ to denote the concepts respectively given by $\top(x) = 1$ and $\bot(x) = 0$ for all $x \in \mathcal{B}^n$.

Borrowing the terminology of computational learning theory (Kearns and Vazirani 1994), a representation class (or language) for a concept class $\mathcal{F}$ is a set of strings $\mathcal{R}$ defined over some (possibly infinite) alphabet of symbols. $\mathcal{R}$ is associated with two surjective functions, namely, a mapping $\llbracket \cdot \rrbracket : \mathcal{R} \rightarrow \mathcal{F}$, called representation scheme, and a mapping $\cdot \rightarrow \mathcal{R} : \mathcal{F} \rightarrow \mathcal{R}$, capturing the size of each representation. Any string $\rho \in \mathcal{R}$ for which $\llbracket \rho \rrbracket = f$ is called a representation of the concept $f$.

A wide spectrum of representation classes have been proposed in the literature for encoding Boolean functions in a compact way. Among them, propositional languages are defined over a set $X_n = \{x_1, \cdots , x_n\}$ of Boolean variables, the constants 1 (true) and 0 (false), and the Boolean connectives $\neg$ (negation), $\lor$ (disjunction) and $\land$ (conjunction). A literal is a variable $x_i$ or its negation $\neg x_i$ (also denoted $\pi_i$), a term or monomial is a conjunction of literals, and a clause is a disjunction of literals. In such a setting, a vector $x \in \mathcal{B}^n$ is also viewed as a term $\bigwedge_{i=1}^n \ell_i$, where for each $i \in [n]$, $\ell_i = x_i$ if the $i$th coordinate of $x$ is 1, and $\ell_i = \pi_i$ if the $i$th coordinate of $x$ is 0. A CNF formula is a finite conjunction of clauses.

For propositional languages, the representation scheme is defined according to the standard semantics of propositional logic. As an example, for the concept class of monomials, each representation is a term $t = \ell_1 \land \cdots \land \ell_k$, and the corresponding concept is:

$$t(x) = \prod_{j=1}^k \ell_j(x)$$

where

$$\ell_j(x) = x_j \quad \text{if } \ell_j = x_j,$$
$$\ell_j(x) = 1 - x_j \quad \text{if } \ell_j = \pi_j.$$

As an alternative to propositional languages conveying a logical interpretation of Boolean functions, neural representation languages are endowed with a geometrical interpretation of concepts (Anthony 2001). The simplest neural representation language is the family of linear threshold functions of the form $f = (w, \tau) \in \mathbb{R}^{n+1}$. For this language, the representation scheme maps $f$ to the concept:

$$f(x) = \mathbb{I}[w_1 x_1 + \cdots + w_n x_n \geq \tau]$$

2 Formal Preliminaries

For a positive integer $n$, let $[n]$ denote the set $\{1, \cdots , n\}$. Let $\mathcal{F}_n$ be the set of all Boolean functions from $\mathcal{B}^n$ into $\mathcal{B}$, where $\mathcal{B} = \{0, 1\}$. Any member $f$ of $\mathcal{F}_n$ is called a concept, and any subset $\mathcal{F}$ of $\mathcal{F}_n$ is called a concept class. Any vector $x$ in the Boolean hypercube $\mathcal{B}^n$ is called an instance; $x$ is a positive example (or model) of some concept $f$ if $f(x) = 1$, and $x$ is a negative example of $f$ if $f(x) = 0$. In what follows, we use $\top$ and $\bot$ to denote the concepts respectively given by $\top(x) = 1$ and $\bot(x) = 0$ for all $x \in \mathcal{B}^n$.

Borrowing the terminology of computational learning theory (Kearns and Vazirani 1994), a representation class (or language) for a concept class $\mathcal{F}$ is a set of strings $\mathcal{R}$ defined over some (possibly infinite) alphabet of symbols. $\mathcal{R}$ is associated with two surjective functions, namely, a mapping $\llbracket \cdot \rrbracket : \mathcal{R} \rightarrow \mathcal{F}$, called representation scheme, and a mapping $\cdot \rightarrow \mathcal{R} : \mathcal{F} \rightarrow \mathcal{R}$, capturing the size of each representation. Any string $\rho \in \mathcal{R}$ for which $\llbracket \rho \rrbracket = f$ is called a representation of the concept $f$.

A wide spectrum of representation classes have been proposed in the literature for encoding Boolean functions in a compact way. Among them, propositional languages are defined over a set $X_n = \{x_1, \cdots , x_n\}$ of Boolean variables, the constants 1 (true) and 0 (false), and the Boolean connectives $\neg$ (negation), $\lor$ (disjunction) and $\land$ (conjunction). A literal is a variable $x_i$ or its negation $\neg x_i$ (also denoted $\pi_i$), a term or monomial is a conjunction of literals, and a clause is a disjunction of literals. In such a setting, a vector $x \in \mathcal{B}^n$ is also viewed as a term $\bigwedge_{i=1}^n \ell_i$, where for each $i \in [n]$, $\ell_i = x_i$ if the $i$th coordinate of $x$ is 1, and $\ell_i = \pi_i$ if the $i$th coordinate of $x$ is 0. A CNF formula is a finite conjunction of clauses.

For propositional languages, the representation scheme is defined according to the standard semantics of propositional logic. As an example, for the concept class of monomials, each representation is a term $t = \ell_1 \land \cdots \land \ell_k$, and the corresponding concept is:

$$t(x) = \prod_{j=1}^k \ell_j(x)$$

where

$$\ell_j(x) = x_j \quad \text{if } \ell_j = x_j,$$
$$\ell_j(x) = 1 - x_j \quad \text{if } \ell_j = \pi_j.$$

As an alternative to propositional languages conveying a logical interpretation of Boolean functions, neural representation languages are endowed with a geometrical interpretation of concepts (Anthony 2001). The simplest neural representation language is the family of linear threshold functions of the form $f = (w, \tau) \in \mathbb{R}^{n+1}$. For this language, the representation scheme maps $f$ to the concept:

$$f(x) = \mathbb{I}[w_1 x_1 + \cdots + w_n x_n \geq \tau]$$

Table 1: Some XAI queries.
where \( \mathbb{1}[p] = 1 \) if \( p \) is true, and \( \mathbb{1}[p] = 0 \) if \( p \) is false. These threshold units can be further generalized to feedforward neural networks, examined at the end of the next section.

Let \( \mathcal{R} \) be a representation language, and \( \mathbb{1}[\cdot] \) a representation scheme mapping \( \mathcal{R} \) into some concept class \( \mathcal{F} \). For a representation \( \rho \in \mathcal{R} \), we use \( \text{Var}(\rho) \) to denote the set of Boolean variables occurring in \( \rho \). The set of models of \( \rho \), given by \( \mathbb{1}[\rho]^{-1}(1) \) is denoted \( \text{mods}(\rho) \). Whenever \( \alpha \) belongs to \( \text{mods}(\rho) \), one also writes \( \alpha \models \rho \). Two representations \( \rho \) and \( \rho' \) of \( \mathcal{R} \) are said to be equivalent, denoted \( \rho \equiv \rho' \), if \( \mathbb{1}[\rho] = \mathbb{1}[\rho'] \). We also say that \( \rho \) entails \( \rho' \), denoted \( \rho \models \rho' \), if \( \text{mods}(\rho) \subseteq \text{mods}(\rho') \). A representation \( \rho \) is inconsistent if \( \mathbb{1}[\rho] = \perp \) and valid if \( \mathbb{1}[\rho] = \top \).

### 3 Boolean Classifiers

Based on elementary notions given in the previous section, we will focus on the concept class \( \mathcal{F} = \mathcal{F}_n \). In other words, all representation languages examined in this study are expressive enough to cover any Boolean function over \( n \) variables. In what follows, a Boolean classifier is simply a representation of some concept in \( \mathcal{F}_n \), according to some representation language \( \mathcal{R} \), associated with its representation scheme and its size measure.

For illustration, the following toy example will be used throughout the paper as a running example:

**Example 1.** The focus is laid on the concept of common hollyhocks (alias alcea rosea). One needs a Boolean classifier to characterize it, i.e., to separate common hollyhocks from other roses using the following four features: \( x_1 \): “has deciduous foliage”, \( x_2 \): “has heart-shaped leaves”, \( x_3 \): “has large flowers”, and \( x_4 \): “has a light green stem”. The concept \( f \in \mathcal{F}_4 \) of common hollyhocks is given by the set of its positive instances \( \{(1,0,1,1), (1,1,0,0), (1,1,1,0), (1,1,1,1)\} \).

**DNF formulae.** Arguably, the simplest language for representing in intuitive terms any Boolean function is the class of DNF formulae, which has been extensively studied in machine learning (Valiant 1985; Pitt and Valiant 1988; Feldman 2009). A DNF formula is a finite disjunction of monomials \( \mathbb{D} = t_1 \lor t_2 \lor \cdots \lor t_m \), and its associated concept is \( \mathbb{D}(\mathbb{x}) = \max_{1 \leq i \leq m} t_i(\mathbb{x}) \). As usual, the size of a DNF formula is defined by the sum of sizes of its terms, where the size of a term is simply given by the number of its literals.

**Example 2.** The concept of common hollyhocks can be represented by:

\[
\mathbb{D} = (x_1 \land x_2 \land \bar{x}_3) \lor (x_1 \land x_2 \land x_3 \land x_4) \\
\lor (x_1 \land x_2 \land x_4) \lor (x_1 \land \bar{x}_2 \land x_3 \land x_4)
\]

**Decision Lists.** The aforementioned DNF formulae can be generalized to rule models, which have received a great deal of attention in the literature of machine learning and knowledge discovery (see e.g. Flach 2012; Fürnkranz et al. 2012) for general surveys). Notably, decision lists (Rivest 1987) are ordered multi-sets of rules of the form \( \mathbb{L} = \langle t_1, c_1 \rangle, \ldots, \langle t_m, c_m \rangle \), where each \( t_i \) \( (i \in [m]) \) is a term over \( X_n \), and each \( c_i \) is a Boolean value in \( \mathbb{B} \). An input instance \( \mathbb{x} \in \mathbb{B}^n \) is a model of \( \mathbb{L} \) if the class \( c_i \) of the first rule \( t_i \) that is matched on \( \mathbb{x} \) is positive. By convention, the last rule \( t_m \) is the empty term \( \top \). Formally, \( \mathbb{L}(\mathbb{x}) = c_j \) where \( j = \text{argmin}_{1 \leq i \leq m} t_i(\mathbb{x}) \). The size of a decision list \( \mathbb{L} \) is the sum of the sizes of the terms occurring in \( \mathbb{L} \).

**Example 3.** The concept of common hollyhocks can be represented by \( \mathbb{L} = \langle x_1 \land x_2, 1 \rangle, \langle \bar{x}_1, 0 \rangle, \langle x_3 \land x_4, 1 \rangle, \langle \top, 0 \rangle \).

**Decision Trees.** Tree models are among the most popular representations in machine learning. In particular, decision trees (Breiman et al. 1984; Quinlan 1986) are models of paramount importance in XAI, as they can be easily read by recursively breaking a choice into sub-choices until a decision is reached. Formally, a (Boolean) decision tree is a binary tree \( T \), where each internal node is labeled with a Boolean variable in \( X_n \), and each leaf is labeled 0 or 1. Without loss of generality, every variable is assumed to appear at most once on any root-to-leaf path (this is called the read-once property). The value \( T(\mathbb{x}) \) of an input instance \( \mathbb{x} \in \mathbb{B}^n \) is given by the leaf reached from the root as follows: for each internal node labeled by \( x_i \), go to the left or right child depending on whether the corresponding value \( x_i \) of \( \mathbb{x} \) is 0 or 1, respectively. The size of \( T \) is given by the number of its nodes.

**Example 4.** The concept of common hollyhocks can be represented by the decision tree \( T \) in Figure 1.

**Random Forests.** Tree models can be generalized to tree ensembles, using ensemble learning techniques, such as bagging and boosting. Notably, the random forest method generates multiple decision trees according to a variant of bagging (Breiman 1996; Breiman 2001). The output representation is a multi-set \( \mathbb{F} = \{T_1, \ldots, T_m\} \) of decision trees, and the corresponding concept is given by:

\[
\mathbb{F}(\mathbb{x}) = \begin{cases} 
1 & \text{if } \sum_{i=1}^m T_i(\mathbb{x}) > \frac{m}{2} \\
0 & \text{otherwise} \end{cases}
\]

In other words, an input instance \( \mathbb{x} \) is a model of \( \mathbb{F} \) if and only if a strict majority of trees in \( \mathbb{F} \) classifies \( \mathbb{x} \) as a positive example. The size of \( \mathbb{F} \) is defined by the sum of sizes of the decision trees occurring in \( \mathbb{F} \).
Example 5. The concept of common hollyhocks can be represented by the random forest in Figure 2.

Boosted Trees. Tree ensembles can also be trained using the boosting technique (Schapire and Schapire 1990; Freund and Schapire 2012) in order to yield boosted trees, which are multi-sets of the form \( B = \{ (T_1, \alpha_1), \ldots, (T_m, \alpha_m) \} \), where each \( T_i (i \in [m]) \) is a decision tree and \( \alpha \) is a convex combination of coefficients.\(^2\) By analogy with random forests, the decisions made by boosted trees are the arcs of the digraph describing the signal flows between nodes, with the arcs of a directed graph corresponding to the presence of threshold units presented above. A neural network is formed when we place units at the vertices of a directed graph, with the depth, width, and size of the neural network given by \( d, \max |V_i|, |V| \), respectively. In a Boolean multilayer perceptron, also known as Boolean multilayer threshold network \( P \) (Anthony 2001), the inputs are vectors in the hypercube \( \mathbb{B}^n \), and the output layer consists of a single node for which the output, denoted \( P(x) \), is a Boolean value in \( \mathbb{B} \).

Example 7. The concept of common hollyhocks can be represented by the Boolean multilayer perceptron \( P \) in Figure 4. The weight of each edge is associated with the corresponding edge. The bias associated with each neuron in layer 2 is \(-1\), and the bias associated with the unique neuron in layer 3 is \(-3\). For the sake of readability, the corresponding inputs are not represented explicitly, but the bias associated with a neuron is written in the box representing the neuron in the figure.

Boolean Multilayer Perceptrons. Based on the linear threshold units presented above, a neural network is formed when we place units at the vertices of a directed graph, with the arcs of a directed acyclic graph \( (V, E) \), and a weight function over the edges: \( w : E \to \mathbb{R} \). Each node \( v \in V \) of the graph captures a neuron. In a multilayer neural network, the set of nodes is decomposed into a union of (nonempty) disjoint subsets \( V = \bigcup_{i=1}^d V_i \), such that the edges in \( E \) connect every node in \( V_l \) to every node in \( V_{l+1} \), for each \( l \in [d-1] \). Accordingly, every neuron \( v \in V \) corresponds to a pair \( l, i \) where \( l \in [d] \) is a layer, and \( i \in |V_l| \) is a rank in layer \( l \). The bottom layer \( V_1 \) is called the input layer and contains \( n \) vertices. The layers \( V_2, \ldots, V_{d-1} \) are called hidden layers, and the top layer \( V_d \) is called the output layer. The inputs of the \( i \)th neuron of the \( l \)th layer with \( 1 < l < d \) are the outputs of all the neurons from layer \( l-1 \), plus an additional input \( b_{l,i} \in \mathbb{R} \), called the bias. We denote by \( o_{l,i}(x) \) the output of the \( i \)th neuron of the \( l \)th layer when the network is fed with the data instance \( x \in \mathbb{R}^n \). With this notation in hand, a multilayer neural network is recursively specified as follows:

\[
o_{l,i}(x) = x_i \quad \text{if } \sum_{i=1}^m \alpha_i T_i(x) > \frac{1}{2}
\]

where \( \text{sgn} \) is the sign function such that \( \text{sgn}(z) = 1[z \geq 0] \). The depth, width, and size of the neural network are given by \( d, \max |V_i|, |V| \), respectively. In a Boolean multilayer perceptron, also known as Boolean multilayer threshold network \( P \) (Anthony 2001), the inputs are vectors in the hypercube \( \mathbb{B}^n \), and the output layer consists of a single node for which the output, denoted \( P(x) \), is a Boolean value in \( \mathbb{B} \).

Binarized Neural Networks. Introduced in (Hubara et al. 2016), binarized neural networks are multilayer neural networks whose activations and weights are predominantly binary (but ranging in \([-1, 1]\)). A BNN is usually described in terms of composition of \( d \) blocks of layers (that are assembled sequentially) rather than individual layers. Thus, a BNN \( N \) consists of a number (say, \( n = d - 1 \)) of internal blocks, followed by a unique output block, noted \( O \). Each block consists of a collection of linear and non-linear transformations. The \( k \)th internal block \( BLK_k (k \in [m]) \) in a BNN can be modeled as a mapping

\[
BLK_k : \{-1, 1\}^{n_k} \to \{-1, 1\}^{n_{k+1}}
\]

associating with a vector of \( n_k \) values in \([-1, 1]\) a vector of \( n_{k+1} \) values in \([-1, 1]\). The inputs of \( BLK_1 \) are the inputs of \( N \) (thus, \( n_1 = n \), the number of elements in \( X_n \), the outputs of \( BLK_k (k \in [m - 1]) \) are the inputs of \( BLK_{k+1} \), the output of \( BLK_m \) is the input of \( O \), and the output value of \( O \) is the output of \( N \). While the input and output of \( N \) are binary vectors, the internal layers of each internal block can

---

\(^2\)In other words, \( \alpha_i \geq 0 \) for all \( i \in [m] \) and \( \sum_i \alpha_i = 1 \).
produce real-valued intermediate outputs. A common construction of an internal block $BLK_k (k \in [m])$ is composed of three main operations:

1. linear transformation (LIN):
   $$y = A_k x_k + b_k,$$
   where $A_k \in \{-1, 1\}^{n_{k+1} \times n_k}$ and $b_k \in \mathbb{R}^{n_{k+1}}$

2. batch normalization (BN):
   $$z_i = \alpha_k \left(\frac{y_i - \mu_k}{\nu_k}\right) + \gamma_k,$$
   where $y = (y_1, y_2, \ldots, y_{n_{k+1}}), \alpha_k, \mu_k, \nu_k, \gamma_k \in \mathbb{R}$, and $\nu_k > 0$

3. binarization (BIN):
   $$x_{k+1} = sgn(z),$$
   where $z = (z_1, z_2, \ldots, z_{n_{k+1}}) \in \mathbb{R}^{n_{k+1}}$,
   and for each $i \in [n_{k+1}], sgn(z_i) = 1$ if $z_i \geq 0$ and $sgn(z_i) = -1$ if $z_i < 0$.
   so that $x_{k+1} \in \{-1, 1\}^{n_{k+1}}$

The output block produces the classification decision. It consists of two layers:

1. linear transformation (LIN):
   $$w = A_d x_d + b_d,$$
   where $A_d \in \{-1, 1\}^{s \times n_d}$ and $b_d \in \mathbb{R}^s$

2. argmax layer (ARGMAX), which outputs the largest index of the largest entry in $w$ as the predicted label
   $$o = \text{argmax}_{i \in [s]} |w_i|$$

Since we are interested in Boolean classification, we suppose that the number of output values of $N$ is $s = 2$ and that for any $x \in \{-1, 1\}^n$, $N$ classifies $x$ as a positive instance if and only if $w_2 > w_1$ (the value of $o$ is 2 in this case, and 1 otherwise).

[Diagram of a boosted tree representation of the concept of common hollyhocks.]

**Example 8.** The concept of common hollyhocks can be represented by the following BNN $N$, with $d = 2$ blocks. We consider only one internal block, with four inputs and five outputs. The parameters of $N$ are defined as follows:

**LIN:**
$$A_1 = \begin{pmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix},$$

**BN:**
$$b_1 = (-3.5, -3.5, -3.5, -3.5, -3.5)$$

**The output block $O$ is defined by:**
$$A_d = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}, b_d = (-4.5, 5)$$

### 4 XAI Queries as Computation Problems

In this section, we consider successively the 9 XAI queries from (Audemard, Koriche, and Marquis 2020), as listed in Table 1, and we present them in formal terms.

**EMC: Enumerating Minimum-Cardinality explanations**

Given an input $x$ such that $\rho(x) = c$, a minimum-cardinality explanation (Shih, Choi, and Darwiche 2018) of $x$ is an instance $x'$ such that $\rho(x') = c$. $x'$ coheres with $x$ on the ones in the sense that for any $k \in \{1, \ldots, n\}$, if $x'_k = 1$ then $x_k = 1$, and $x'$ has a minimal number of coordinates set to 1. Roughly speaking, the features that are set to 1 in $x'$ are enough to explain why $x$ has been classified by $\rho$ as a positive (or as a negative) instance. Formally:

**Definition 1 (EMC).** EMC can be stated as the following problem:

- **Input:** A Boolean representation $\rho$ over $X_n$ and an instance $x \in \mathbb{B}^n$.
- **Output:** Enumerate with polynomial delay the set of all minimum-cardinality explanations of $x$ given $\rho$.

The number of minimum-cardinality explanations of $x$ given $\rho$ can be exponential in the size of the input, thus the time needed to compute all of them is provably exponential as well. EMC[1] denotes the relaxation of EMC where the output consists of a single minimum-cardinality explanation of $x$ given $\rho$. Considering any Boolean classifier $\rho$ for Example 1, $(1, 1, 0, 0)$ is the output of EMC[1] for input $\rho$ and $x = (1, 1, 1, 1)$. [Diagram of a Boolean multilayer perceptron representation of the concept of common hollyhocks.]
**DPI: Deriving one Prime Implicant explanation** Given an input $x$ such that $\rho(x) = c$, a prime implicant explanation of $x$ (Shih, Choi, and Darwiche 2018) (also referred to as a sufficient reason for $x$ given $\rho$ (Darwiche and Hirth 2020)) is a subset-minimal partial assignment $x'$ which is coherent with $x$ (i.e., $x$ and $x'$ give the same values to the variables that are assigned in $x'$) and which satisfies the property that for every extension $x''$ of $x'$ over $X_n$, we have $\rho(x'') = c$. The features assigned in $x'$ (and the way they are assigned) can be viewed as explaining why $x$ has been classified by $\rho$ as a positive (or as a negative) instance. Formally:

**Definition 2 (DPI).** DPI can be stated as the following problem:

- **Input:** A Boolean representation $\rho$ over $X_n$ and an instance $x \in \mathbb{B}^n$.
- **Output:** A prime implicant explanation of $x$ given $\rho$.

Considering any Boolean classifier $\rho$ for Example 1, $x_1 \land x_2$ can be got as an output of DPI for input $\rho$ and $x = (1, 1, 1, 1)$.

**ECO: Enumerating Counterfactual explanations** Counterfactual explanations are required when the user is surprised by the result $y$ provided by the classifier $\rho$ on a given instance $x$. We have $\rho(x) = 1$ (resp. $= 0$) while the user was expecting $\rho(x) = 0$ (resp. $= 1$). A counterfactual explanation of $x$ given $\rho$ is an instance $x'$ which is as close as possible to $x$ in terms of Hamming distance and such that $\rho(x') \neq \rho(x)$. If there is no $x'$ such that $\rho(x') \neq \rho(x)$, then no counterfactual explanation of $x$ given $\rho$ exists. When $x'$ exists, the set of features that differ in $x$ and $x'$ can be viewed as an explanation as to why $x$ has not been classified as expected by $\rho$. Formally:

**Definition 3 (ECO).** ECO can be stated as the following problem:

- **Input:** A Boolean representation $\rho$ over $X_n$ and an instance $x \in \mathbb{B}^n$.
- **Output:** Enumerate with polynomial delay the set of all counterfactual explanations of $x$ given $\rho$.

The number of counterfactual explanations of $x$ given $\rho$ can be exponential in the size of the input, thus the time needed to compute all of them is provably exponential as well. ECO[1] denotes the relaxation of ECO where the output consists of a single counterfactual explanation of $x$ given $\rho$ when such an explanation exists, and $\emptyset$ otherwise. Considering any Boolean classifier $\rho$ for Example 1, $(0, 1, 1, 1)$ is the output of ECO[1] for input $\rho$ and $x = (1, 1, 1, 1)$.

**CIN: Counting the Instances associated with a given class** Counting the number of instances associated with the given class corresponding to $\rho$ is a useful verification query. When the number found heavily differs from the expected one, this may reflect an issue with the dataset used to learn the parameters of the classifier. Formally:

**Definition 4 (CIN).** CIN can be stated as the following problem:

- **Input:** A Boolean representation $\rho$ over $X_n$, and a target class $c \in \mathbb{B}$ (positive or negative instances).
- **Output:** The number of instances $x \in \mathbb{B}^n$ classified by $\rho$ as positive instances if $c = 1$, or as negative instances if $c = 0$.

Considering any Boolean classifier $\rho$ for Example 1, 11 is the output of CIN for input $\rho$ and $c = 0$.

**EIN: Enumerating the Instances associated with a given class** EIN is the enumeration problem that corresponds to CIN:

**Definition 5 (EIN).** EIN can be stated as the following problem:

- **Input:** A Boolean representation $\rho$ over $X_n$, and a target class $c \in \mathbb{B}$ (positive or negative instances).
- **Output:** Enumerate with polynomial delay the set of positive instances $x \in \mathbb{B}^n$ according to $\rho$ if $c = 1$ and the set of negative instances $x \in \mathbb{B}^n$ according to $\rho$ if $c = 0$.

The number of positive (or negative) instances $x \in \mathbb{B}^n$ according to $\rho$ can be exponential in the size of the input, thus the time needed to compute all of them is provably exponential as well. EIN[1] denotes the relaxation of EIN where the output consists of a single instance. Considering any Boolean classifier $\rho$ for Example 1, $(0, 1, 1, 1)$ can be got as an output of EIN[1] for input $\rho$ and $c = 0$.

**IMA: Identifying Mandatory features / forbidden features in a given class** When the frequency of a feature $x_k$ (or combination of features) in the class of positive (or negative) instances associated with $\rho$ is equal to 1, the feature / combination of features is mandatory for the class of positive (or negative) instances (as they are perceived by the classifier) is useful (the classifier should be such that there is no discrepancy between what is got and what was expected). Formally:

**Definition 6 (IMA).** IMA can be stated as the following problem:

- **Input:** A Boolean representation $\rho$ over $X_n$, a term $t$ over $X_n$, and a target class $c \in \mathbb{B}$ (positive or negative instances).
- **Output:** 1 if $t$ is mandatory for the class of positive (resp. negative) instances when $c = 1$ (resp. $c = 0$), and 0 otherwise.

A similar definition can be stated for forbidden features. Considering any Boolean classifier $\rho$ for Example 1, 1 is the output of IMA for input $\rho$, $t = x_1$, and $c = 1$.

**IIR: Identifying Irrelevant features in a given class** A feature $x_i$ is irrelevant for the class of positive (resp. negative) instances associated with $\rho$ if and only if for every positive (resp. negative) instance $x$ according to $\rho$, the instance $x'$ that coincides with $x$ on every feature but $x_i$ is also classified positively (resp. negatively) by $\rho$. Deciding
whether a feature is irrelevant or not for the class of positive (resp. negative) instances associated with \( \rho \) is a useful verification query for identifying decision bias: there is such a bias when the membership of any instance \( x \) to the class associated with \( \rho \) depends on its value for the feature \( x_i \) while it should not. Formally:

**Definition 7 (IIR).** IIR can be stated as the following problem:

- **Input:** A Boolean representation \( \rho \) over \( X_n \), a feature \( x_i \in X_n \), and a target class \( c \in \mathbb{B} \) (positive or negative instances).
- **Output:** 1 if \( x_i \) is irrelevant for the class of positive (resp. negative) instances associated with \( \rho \) when \( c = 1 \) (resp. \( c = 0 \)), and 0 otherwise.

Considering any Boolean classifier \( \rho \) for Example 1, 0 is the output of IIR for input \( \rho \) and \( c = 1 \), whatever \( x_i \) (\( i \in [4] \)).

**IMO: Identifying MOnotone (or anti-monotone) features in a given class.** In many applications, it is believed that increasing the value of some feature does not change the membership to the class of positive (resp. negative) instances associated with the Boolean classifier. Dually, one might also expect that decreasing the value of some other feature does not change the membership to the class. It is important to be able to test whether the classifier \( \rho \) that has been generated complies or not with such beliefs.

Making it formal calls for a notion of monotonicity (or anti-monotonicity) of a classifier, which can be stated as follows: a classifier \( \rho \) is monotone (resp. anti-monotone) with respect to an input feature \( x_i \) for the class of positive (resp. negative) instances, if for any positive (resp. negative) instance \( x \) according to \( \rho \), we have \( \rho(x[x_i \leftarrow 1]) = 1 \) (resp. \( \rho(x[x_i \leftarrow 0]) = 1 \)). Formally:

**Definition 8 (IMO).** IMO can be stated as the following problem:

- **Input:** A Boolean representation \( \rho \) over \( X_n \), a feature \( x_i \in X_n \), and a target class \( c \in \mathbb{B} \) (positive or negative instances).
- **Output:** 1 if \( \rho \) is monotone (resp. anti-monotone) w.r.t. \( x_i \) for the class of positive (resp. negative) instances, 0 otherwise.

Considering any Boolean classifier \( \rho \) for Example 1, 1 is the output of IMO for input \( \rho \) and \( c = 1 \), whatever \( x_i \) (\( i \in [4] \)).

**MCP: Measuring Closeness of a class to a Prototype.** Finally, one can also be interested in determining how much a given prototype \( x \) complies with the class that the classifier \( \rho \) associates with it. This can be evaluated by computing the Hamming distance between \( x \) and every element of \( \{x' \in \mathbb{B}^n : \rho(x') = \rho(x)\} \) and considering the maximal distance. When a prototype of a class exists, it is supposed to be a “central” element of the class (i.e., minimizing the maximal distance to any other element of the class). Thus, a large value may indicate a problem with the classifier that has been learned. Formally:

**Definition 9 (MCP).** MCP can be stated as the following problem:

- **Input:** A Boolean representation \( \rho \) over \( X_n \) and an instance \( x \in \mathbb{B}^n \).
- **Output:** The maximal Hamming distance of \( x \) to the class of positive (resp. negative) instances when \( \rho(x) = 1 \) (resp. \( \rho(x) = 0 \)).

Considering any Boolean classifier \( \rho \) for Example 1, 2 is the output of MCP for input \( \rho \) and \( x = (1, 1, 1, 1) \).

## 5 On the Intelligibility of XAI Queries

We are now in position to evaluate the computational intelligibility of each family of classifiers, among decision trees, \( \text{DNF} \) classifiers, decision lists, random forests, boosted trees, Boolean multilayer perceptrons, and binarized neural nets over Boolean features. This intelligibility is assessed by determining the set of XAI queries (out of the 9 ones considered in the previous section) that are offered by each family, i.e., those for which the corresponding computation problem is tractable.

Since the computation problems associated with XAI queries are not always decision problems, the intractability of a computation problem is established by proving that it is \( \text{NP} \)-hard in the sense of Cook reduction; in this case, the existence of a (deterministic) polynomial-time algorithm to solve the corresponding XAI query would imply that \( \mathcal{P} = \text{NP} \), giving thus strong evidence that such an algorithm does not exist.

The main results of the paper are synthesized in the two following propositions:

**Proposition 1.** For each enumeration problem among EMC, ECO, EIN, there exists an enumeration algorithm with polynomial delay when the Boolean classifier under consideration is a decision tree over \( X_n \). Furthermore, each problem among DPI, CIN, IMA, IIR, IMO, MCP is in \( \mathcal{P} \) when the Boolean classifier under consideration is a decision tree over \( X_n \).

**Proof.** By definition, for each of the 9 XAI queries, the target class can be the one of positive instances or the one of negative instances. This does not raise any issue for decision trees. Indeed, for any decision tree \( T \) over \( X_n \), one can compute in linear time a decision tree \( T' \) representing the complementary class to the one associated with \( T \), i.e., a decision tree \( T' \) such that \( \forall x \in \mathbb{B}^n, T'(x) = 1 \) if and only if \( T'(x) = 0 \). To get \( T' \) from \( T \), it is enough to replace in \( T \) every 1-leaf node by a 0-leaf node, and every 0-leaf node by a 1-leaf node.\(^4\)

Now, (Audemard, Koriche, and Marquis 2020) have identified sufficient conditions for a (multi-label, yet Boolean) classifier to offer XAI queries based on the queries and transformations of the language \( \mathcal{L} \) used to represent it. Those

\(^4\)Stated otherwise, \( \text{DT} \) satisfies the \( \neg \mathcal{C} \) transformation from the knowledge compilation map (Koriche et al. 2013).
queries and transformations are standard queries and transformations from the knowledge compilation map (Darwiche and Marquis 2002).

It turns out that the language $\mathcal{D}$ of decision trees over Boolean variables satisfies many of those queries and transformations, namely CO, CD, ME, CT, IM, OPT, EQ, SE. This has been shown in (Koriche et al. 2013) for all of them, but OPT. As to OPT, it is easy to adapt the proof that $\mathcal{DNF}$ satisfies OPT (Darwiche and Marquis 2004; Koriche et al. 2016) to the case of $\mathcal{D}$. Indeed, let $w_v \in \mathbb{Q}$ be a number (the weight of $v \in X_n$). For any interpretation $x$ over $X_n$, one defines $f_w(x) = \sum_{v \in X_n} w_v \cdot x(v)$. Now, for any formula $\varphi$, one defines $f_w(\varphi) = \min\{f_w(x) : x \models \varphi\}$.5 It is easy to show by structural induction that when $\alpha = T$ is a decision tree over $X_n$, $f_w(T)$ can be computed in polynomial time in the size of $T$ when all the weights $w_v$ are bounded by a constant that does not depend on $T$ (which is a reasonable assumption). Indeed, we have $f_w(0) = \infty$, $f_w(1) = 0$, and

$$f_w(ite(v, T_1, T_2)) = w_v + \min\{f_w(T_1), f_w(T_2)\}.$$

On this ground, starting from $T$, one can generate in polynomial time a decision tree $opt(T)$ over $\varphi(T)$ the models of which being precisely the models of $T$ over $\varphi(T)$, that minimize the value of $f_w$. Indeed, we have $opt(0) = 0$, $opt(1) = 1$, and

$$opt(ite(v, T_1, T_2)) = ite(v, opt(T_1), opt(T_2)) \text{ if } f_w(T_1) = f_w(T_2)$$

$$= ite(v, opt(T_1), 0) \text{ if } f_w(T_1) < f_w(T_2)$$

$$= ite(v, 0, opt(T_2)) \text{ if } f_w(T_1) > f_w(T_2)$$

Then using results reported in (Audemard, Koriche, and Marquis 2020), we get that $\mathcal{D}$ offers the XAI queries EMC, ECO, CIN, EIN, IMA, MCP. Finally, though $\mathcal{D}$ does not satisfy FO (Koriche et al. 2013), the XAI queries that have been addressed using forgetting in (Audemard, Koriche, and Marquis 2020) require to apply the forgetting transformation to eliminate from $T$ variables representing classes. This is useless here since no class variable is used in $T$ (only two classes are implicitly considered here, the one associated with $T$, alias the class of positive instances, and its complementary set which can be obtained by computing $\neg T$). Thus, $\mathcal{D}$ also offers the XAI queries DPI, IIR, and IMO.

\[\Box\]

Proposition 2. Each problem among EMC[1], DPI, ECO[1], CIN, EIN[1], IMA, IIR, IMO, MCP where the Boolean classifier under consideration $\rho$ is given as a CNF formula.

In a second part, we show how a CNF classifier $\rho$ can be associated in polynomial time with an equivalent classifier having the form of a decision list, a random forest, a boosted tree, a Boolean multilayer perceptron, or a binarized neural net over Boolean features.

Combining the polynomial reductions from the first part with the polynomial translations of the second part, the NP-hardness results stated in the proposition and concerning decision lists, random forests, boosted trees, Boolean multilayer perceptrons, and binarized neural nets follow. Finally, the case of $\mathcal{DNF}$ classifiers is addressed in a third part.

Let us start with the first part of the proof:

- **EMC[1]**. Let $\alpha = \bigwedge_{i=1}^k \delta_i$ be a CNF formula over $\{x_1, \ldots, x_{n-1}\}$. We associate with $\alpha$ in polynomial time the ordered pair $(\rho, x)$ where $\rho = \bigwedge_{i=1}^k \bigvee_{j=1}^n (\delta_i \lor x_j)$ is a CNF formula over $X_n = \{x_1, \ldots, x_n\}$ (equivalent to $\alpha \lor (\bigwedge_{i=1}^n x_i)$), and $x = \bigwedge_{i=1}^n x_i$. Clearly enough, $\rho$ classifies $x$ as a positive instance. Now, there are two cases:
  - If $\alpha$ is unsatisfiable, then $\rho$ is equivalent to $\bigwedge_{i=1}^n x_i$.
  - If $\alpha$ is satisfiable, then it has a model $x'$ over $\{x_1, \ldots, x_{n-1}\}$. The instance $x'' \in \mathbb{B}^n$ that extends $x'$ and sets $x_n$ to 0 is classified as a positive instance by $\rho$, and it contains less features set to 1 than $x$, thus $x$ is not a minimum-cardinality explanation of $x$ in this case.

- **DPI**. Let $\alpha$ be a CNF formula over $\{x_1, \ldots, x_{n-1}\}$. We associate with $\alpha$ in polynomial time the ordered pair $(\rho, x)$ where $\rho = \alpha \land (\bigvee_{i=1}^n \neg x_i)$ is a CNF formula over $X_n = \{x_1, \ldots, x_n\}$ and $x = \bigwedge_{i=1}^n x_i$. By construction, $x$ is classified by $\rho$ as a negative instance.

- **ECO[1]**. Consider the same polynomial reduction as in the DPI case. There are two cases:
  - If $\alpha$ is unsatisfiable, then every instance $x' \in \mathbb{B}^n$ is classified by $\rho$ as a negative instance since $\rho$ is equivalent to $\bot$. This is equivalent to state that there is a unique prime implicand explanation of $x$ classified by $\rho$ as a negative instance, namely $T$.
  - If $\alpha$ is satisfiable, then it has a model over $\{x_1, \ldots, x_{n-1}\}$, and the instance $x'' \in \mathbb{B}^n$ that extends this model and sets $x_n$ to 0 is a model of $\alpha \land (\bigvee_{i=1}^n \neg x_i)$. Thus, $x''$ is classified by $\rho$ as a positive instance, and as a consequence $T$ is not a prime implicand explanation of $x$ given $\rho$.

- **IMO**. We set $w_v$ to 0 whenever $v$ does not occur in $\varphi$.

5A more direct proof can be found in (Izza, Ignatiev, and Marques-Silva 2020).
In this case, the set of positive instances given \( \rho \) is not empty, and as a consequence, a counterfactual explanation of \( x \) given \( \rho \) exists.

- **CIN.** Let \( \alpha \) be a CNF formula. We associate with \( \alpha \) in polynomial time the ordered pair \(( \rho, c)\) where \( \rho = \alpha \land x_n \) and \( c = 1 \). The number of instances \( x \in \mathbb{B}^n \) classified positively by \( \rho \) is equal to the number of models of \( \alpha \). If it was possible to compute this number in polynomial time, then one could decide in polynomial time whether \( \alpha \) is satisfiable or not.

- **EIN[1].** We consider the same polynomial reduction as in the CIN case. An instance \( x \in \mathbb{B}^n \) classified by \( \rho \) as a positive instance exists if and only if \( \alpha \) is satisfiable.

- **IMA.** Let \( \alpha \) be a CNF formula over \( \{x_1, \ldots, x_{n-1}\} \). We associate with \( \alpha \) in polynomial time the triple \(( \rho, t, c)\) where \( \rho \) is the same formula as in the proof for the EMC[1] case, \( t = x_n \), and \( c = 1 \):
  - If \( \alpha \) is unsatisfiable, then \( x = \bigwedge_{i=1}^n x_i \) is the unique instance of \( \mathbb{B}^n \) that is classified positively by \( \rho \). Thus, every feature from \( x \) (especially, those of \( t = x_n \)) is mandatory for the class of positive instances.
  - If \( \alpha \) is satisfiable, then it has a model over \( \{x_1, \ldots, x_{n-1}\} \) and the instance \( x' \) that extends this model and sets \( x_n \) to 0 is classified positively by \( \rho \), showing that \( t = x_n \) is not mandatory for the class of positive instances.

The case of forbidden features is similar (consider \( \rho = \alpha \lor \bigvee_{i=1}^n \overline{x_i} \) instead of \( \rho = \alpha \lor \bigvee_{i=1}^n x_i \); \( t \) is forbidden for the class of positive instances if and only if \( \alpha \) is unsatisfiable).

- **IRR.** Let \( \alpha \) be a CNF formula over \( \{x_1, \ldots, x_{n-1}\} \). Let us associate with \( \alpha \) in polynomial time the triple \(( \rho, x_n, c)\) where \( \rho = \alpha \land x_n \) is a CNF formula over \( X_n = \{x_1, \ldots, x_n\} \), and \( c = 0 \):
  - If \( \alpha \) is unsatisfiable, then \( x_n \) is an irrelevant feature for the class of negative instances associated with \( \rho \).
  - If \( \alpha \) is satisfiable, then it has a model over \( \{x_1, \ldots, x_{n-1}\} \). Consider the instance \( x \) that extends this model and sets \( x_n \) to 1. Then \( \rho(x) = 1 \). However the instance \( x' = x \land \overline{x_n} \) that coincides with \( x \) on every feature but \( x_n \) is such that \( \rho(x') = 0 \). This shows that \( x_n \) is relevant for the class of negative instances associated with \( \rho \).

- **IMO.** Let \( \alpha \) be a CNF formula over \( \{x_1, \ldots, x_{n-1}\} \). Let us associate with \( \alpha \) in polynomial time the triple \(( \rho, x_n, c)\) where \( \rho = \alpha \land \overline{x_n} \) is a CNF formula over \( X_n = \{x_1, \ldots, x_n\} \), and \( c = 1 \):
  - If \( \alpha \) is unsatisfiable, then \( x_n \) is unsatisfiable as well and, as such, \( \rho \) is obviously monotone w.r.t. \( x_n \) (it is monotone w.r.t. every feature).
  - If \( \alpha \) is satisfiable, then it has a model over \( \{x_1, \ldots, x_{n-1}\} \). Consider the instance \( x \) that extends this model and sets \( x_n \) to 0. Then \( \rho(x) = 1 \). However the instance \( x' = x \land \overline{x_n} \) that coincides with \( x \) on every feature but \( x_n \) is such that \( \rho(x') = 0 \). This shows that \( \rho \) is not monotone w.r.t. \( x_n \) for the class of positive instances.

The case of anti-monotone features is similar (consider \( \rho = \alpha \lor x_n \) and extends the counter-model of \( \alpha \) by setting \( x_n \) to 1 when \( \alpha \) is satisfiable).

- **MCP.** Let \( \alpha \) be a CNF formula over \( \{x_1, \ldots, x_{n-1}\} \). Consider the same polynomial reduction as in the EMC[1] case:
  - If \( \alpha \) is unsatisfiable, then the unique instance classified positively by \( \rho \) is \( x \), showing that the maximal Hamming distance of \( x \) to an element of the class associated with \( \rho \) is 0.
  - If \( \alpha \) is satisfiable, then it has a model over \( \{x_1, \ldots, x_{n-1}\} \) and the instance \( x' \) that extends this model and is such that \( x_n \) is set to 0 is classified positively by \( \rho \). In this case, the maximal Hamming distance of \( x \) to the class associated with \( \rho \) is \( \geq 1 \).

Let us now present the second part of the proof:

- **Decision lists.** Every CNF formula \( \rho \) can be turned in linear time into an equivalent decision list \( L \) (see Theorem 1 from (Rivest 1987)). Accordingly, every reduction pointed out in the first part of the proof can be turned into a reduction such that the targeted representation is a decision list, and this concludes the proof.

- **Random forests.** We exploit the same idea as in the proof for the decision lists case. To get the result, it is enough to show that every CNF formula \( \rho \) can be turned in linear time into an equivalent random forest \( F \). The translation is as follows: given a CNF formula \( \rho = \bigwedge_{i=1}^k \delta_i \) over \( X_n \), we associate with it in linear time the random forest

\[
F = \{T_1, \ldots, T_k, 0, \ldots, 0\}
\]

over \( X_n \), where each \( T_i \) (i ∈ [k]) is a decision tree over \( X_n \) that represents the clause \( \delta_i \). Each \( T_i \) (i ∈ [k]) is a comb-shaped tree that can easily be generated in time linear in the size of \( \delta_i \); if \( \delta_i \) is an empty clause, then return \( T_i = 0 \), else considering the literals \( \ell \) of \( \delta_i \) in sequence, generate a decision node of the form \( i \ell(x, 1, T_{i,\ell}) \) (resp. \( i \ell(x, T_{i,\ell}^\top, 1) \)) if \( \ell \) is a negative literal \( \overline{x} \) (resp. a positive literal \( x \)), where \( T_{i,\ell}^\top \) is a decision tree for the clause \( \delta_i \setminus \{\ell\} \). Finally, by construction, the only subset of trees of \( F \) that contains more than half of the trees and that can be consistent is \( \{T_1, \ldots, T_k\} \) (every other subset of \( F \) containing at least \( k \) trees contains a tree reduced to 0, and as such, is inconsistent). Accordingly, \( F \) is equivalent to the conjunction of all trees from \( \{T_1, \ldots, T_k\} \). Since each \( T_i \) (i ∈ [k]) is equivalent to the clause \( \delta_i \) of \( \rho \), we get that \( F \) is equivalent to \( \rho \), as expected.

- **Boosted trees.** Direct from the proof for the random forests case, given that a random forest \( F = \{T_1, \ldots, T_m\} \) can be turned in linear time into an equivalent boosted tree \( B = \{\langle T_1, \frac{1}{m}\rangle, \ldots, \langle T_m, \frac{1}{m}\rangle\} \) where the weight of each tree is equal to \( \frac{1}{m} \).
• **Boolean multilayer perceptrons.** It is not difficult to turn in polynomial time any CNF formula $\rho = \bigwedge_{i=1}^{k} \delta_i$ over $X_n = \{x_1, \ldots, x_n\}$, such that $\rho$ does not contain any valid clause (this can be ensured efficiently), into a Boolean multilayer perceptron $P$ over $X_n$, that is logically equivalent to $\rho$. One uses only three layers: as expected, the first one $V_1$ contains $n$ vertices (on per variable $x_i \in X_n$), the last one $V_3$ contains a single vertex $v_{3,1}$, and the second layer $V_2$ contains $k$ vertices, one per clause in $\rho$. The output of $v_{3,1}$ is the output of $P$. Let $\delta_i$ be any clause of $\rho$ and let $v_{2,i}$ be the corresponding vertex. For every vertex $v_{1,j}$ ($j \in [n]$) from the first layer $V_1$ that is associated with a variable $x_j \in X_n$, the edge $(v_{1,j}, v_{2,i}) \in E$ that connects $v_{1,j}$ to $v_{2,i}$ is labelled by $w(v_{1,j}, v_{2,i}) = 0$ if $x_j$ does not occur in $\delta_i$, by $w(v_{1,j}, v_{2,i}) = 1$ if $x_j$ is a positive literal of $\delta_i$, and by $w(v_{1,j}, v_{2,i}) = -1$ if $-x_j$ is a negative literal of $\delta_i$. The value of the bias $b_{2,i}$ is the number of negative literals in $\delta_i$, minus 1. By construction, we have $\omega_{v_{2,i}}(x) = 1$ if and only if $x$ satisfies the clause $\delta_i$ associated with $v_{2,i}$. Now, for every vertex $v_{2,i}$ ($i \in [k]$) of the second layer $V_2$, the edge $(v_{2,i}, v_{3,1}) \in E$ that connects $v_{2,i}$ to $v_{3,1}$ is labelled by $w(v_{2,i}, v_{3,1}) = 1$, and the bias $b_{3,1}$ is set to $-k$. Accordingly, the output of $v_{3,1}$ of $P$ is 1 if and only if every clause $\delta_i$ of $\rho$ is satisfied by $x$, or stated otherwise, if and only if $x$ is a model of $\rho$.

• **Binarized neural nets.** With a CNF formula $\rho = \bigwedge_{i=1}^{k} \delta_i$ over $X_n = \{x_1, \ldots, x_n\}$, such that $\rho$ does not contain any valid clause, we associate in polynomial time a BNN $N$ with $2n$ inputs in $\{-1, 1\}$ and an output value in $\{1, 2\}$ such that for any $x \in \mathbb{B}^n$, $x$ is a model of $\rho$ if and only if its translation $\text{transl}(x) \in \{-1, 1\}^{2n}$ is classified as a positive instance by $N$ (i.e., the output value of $N$ is 2). The translation mapping $\text{transl} : \mathbb{B}^n \rightarrow \{-1, 1\}^{2n}$ can be computed in linear time and is defined as follows:

$$\text{transl}(x_1, \ldots, x_n) = (2x_1 - 1, 2x_1 - 1, \ldots, 2x_n - 1, 2x_n - 1)$$

i.e., for $i \in \{0, \ldots, n-1\}$, the $(2i + 1)^{th}$ coordinate and the $2(i + 1)^{th}$ coordinate of $\text{transl}(x_1, \ldots, x_n)$ are equal to $2x_{i+1} - 1$. $N$ consists of a single intermediate block, i.e., $m = 1$, so that the total number of blocks is $d = 2$. The number of inputs of the first block is $2n$ and the number of outputs of this block is $k$, the number of clauses of $\rho$.

The key idea of our translation from CNF to BNN is to consider each of the $k$ clauses $\delta$ of $\rho$ individually and compute the difference between the number of literals of $\delta$ falsified by $x$ and the number of literals of $\delta$ satisfied by $x$, which is noted as follows:

$$\text{diff}(\delta, x) = |\{\ell \in \delta : x = \top\}| - |\{\ell \in \delta : x = \bot\}|.$$  

Obviously enough, $x$ does not satisfy $\delta$ precisely when $\text{diff}(\delta, x)$ is the number of literals occurring in $\delta$.

The first operation realized by the BNN is a linear transformation. Thus one must define $A_1$ and $b_1$. Basically, one wants to use this transformation to store in the output $y = A_1 \text{transl}(x) + b_1$ some information about the satisfaction of the clauses of $\rho$ by $x$, so that $y_i$ ($i \in [k]$) corresponds to the clause $\delta_i$ of $\rho$. Because clauses are in general not built upon all variables, we need to add a mechanism to avoid considering the variables that do not appear in a clause. This is achieved within $\text{transl}$ by duplicating the coordinates of the input vector $x$ (in addition to translating them from $\mathbb{B}$ to $\{-1, 1\}$). Indeed, if a literal $\ell$ over $x \in X_n$ is not present in a clause $\delta_i$, its contribution to $y_i$ is expected to be 0, which is achieved by multiplying the two coordinates associated with $x$ in $\text{transl}(x)$ by $-1$ and 1, respectively. If a literal $\ell$ over $x \in X_n$ occurs in $\delta_i$, we want its contribution to $y_i$ to be set to 1 if the literal falsifies the clause and to $-1$ if it satisfies the clause. By summing up the contribution of each literal in that way, we obtain the expected result. Formally, the matrix $A_1[i]$ for $i \in [k]$ is defined as:

$$A_1[i] = (\tau_{1i}^1, \tau_{1i}^2, \tau_{1i}^3, \tau_{1i}^4, \ldots, \tau_{1i}^n, \tau_{1i}^{n+1}),$$

for each $j \in [n]$:

$$\begin{align*}
\tau_{1j}^1 & = -\tau_{1j}^2 & \text{if } x_j \notin \text{Var}(\delta_i) \\
\tau_{1j}^3 & = \tau_{1j}^4 & \text{if } x_j \in \delta_i \\
\tau_{1j}^5 & = \tau_{1j}^6 & \text{if } -x_j \in \delta_i \\
\tau_{1j}^7 & = \tau_{1j}^8 & \text{if } x_j \in \delta_i
\end{align*}$$

Then one sets $b_1[i] (i \in [k])$ to $-2 \times |\delta_i| + 1$. Overall, we get that for every $i \in [k]$, $y_i = 1$ if the clause $\delta_i$ is falsified by $x$ and $y_i < 0$ if $\delta_i$ is satisfied by $x$. As to batch normalization, for every $i \in [k]$, we set the parameters $\alpha_{ij} = v_{1j}$, to 1, and $\mu_{ij} = \gamma_{ij}$, to 0, so that the output of the transformation coincides with its input. Finally, the binarization transformation takes place. Clearly enough, the output of the internal block of $N$ is a vector $x' = (x'_1, \ldots, x'_k) \in \{-1, 1\}^k$ such that for every $i \in [k], x'_i = 1$ if $x$ falsifies $\delta_i$ and $x'_i = -1$ if $x$ satisfies $\delta_i$.

By construction, this vector $x'$ is the input of the output block $O$ of $N$. Let us recall that the output value $o$ of $O$ is the output of $N$, it is a value in $\{1, 2\}$ indicating whether or not the input $\text{transl}(x)$ is classified positively by $N$. The linear transformation used in $O$ is given by $A_d = \{-1\}_{2 \times 2k}^{2 \times k}$ and $b_d \in \mathbb{R}^2$ such that $A_d[1] = (-1, \ldots, -1)$ and $A_d[2] = (-1, \ldots, -1)$, while $b_d[1] = 2k$ and $b_d[2] = 1$. As such, when the instance $x$ satisfies $\rho$, it satisfies every clause of it and the coordinate $w_2$ of the output $w$ of the linear transformation $w = A_d x' + b_d$ is equal to $k + 1$, while its coordinate $w_1$ is equal to $k$. Contrariwise, when the instance $x$ does not satisfy $\rho$, we have $w_2 \leq k$ and $w_1 \geq k + 1$. As a consequence, the last transformation of $O$ (the ARGMAX layer) will return $o = 2$ when $x$ satisfies $\rho$ and $o = 1$ otherwise. Thus, $N$ can be viewed as a representation of the CNF formula $\rho$ modulo the translation mapping $\text{transl}$, in the sense $\forall x \in \mathbb{B}^n, \rho(x) = c$ if only if $N(\text{transl}(x)) = c + 1$.

Finally, with each reduction given in the first part of the proof, we can associate another polynomial reduction where $\rho$ is a DNF classifier. This comes from two points: (1) the duality relating CNF classifiers to DNF classifiers stating that $x \in \mathbb{B}^n$ is a positive instance of a concept represented by a CNF classifier $\rho$ over $X_n$ if and only if $x$ is a negative instance of the (complementary) concept represented by a
DNF classifier $D$ that is equivalent to $\neg \rho$ (obviously, such a $D$ is computable from $\rho$ in linear time by applying De Morgan’s laws); and (2) the fact that all the XAI queries we have considered must be able to take account for both positive and negative instances. This concludes the proof. \hfill \square

The last two propositions thus show the existence of a large computational intelligibility gap between the families of classifiers at hand. Since each of the 9 XAI queries is tractable for the family of decision trees, decision trees can be considered as highly intelligible in comparison to the other families of classifiers considered in the paper. At the other extremity of the spectrum, DNF classifiers, decision lists, random forests, boosted trees, Boolean multilayer perceptrons, and binarized neural nets appear as poorly intelligible since none of the 9 XAI queries is tractable for any of those families.

Notably, the results reported in the last two propositions differ significantly from those presented in a number of previous papers where an equivalence-preserving polynomial-time translation (alias an encoding) from a given language $L$ of Boolean classifiers to CNF formulae is looked for (see e.g., (Narodytska 2018; Narodytska et al. 2018; Narodytska et al. 2020)). Determining such translations permits to take advantage of automated reasoning techniques for addressing XAI queries, given that existing solvers for Boolean representations are most of the time based on the CNF format. Here, we have looked for polynomial-time translations from the language of CNF formulae to the languages $L$ of the classifiers we focus on, in order to prove that the XAI queries are computationally hard whenever the input classifier is in $L$. This is quite a different, yet complementary perspective. Indeed, when such translations exist, leveraging CNF encodings to address XAI queries makes sense from a computational standpoint, i.e., it is not using a sledgehammer to crack a nut. However, those translations are not guaranteed to exist for every family $L$ of classifiers, so that it can be the case that an XAI query is tractable for the $L$ while being computationally hard for CNF classifiers. Accordingly, our study shows that this is precisely what happens with decision trees. For this family of classifiers, it is meaningful to develop algorithms for addressing XAI queries that are directly based on the representations at hand (decision trees), instead of designing CNF encodings.

6 Conclusion

In this paper, we have investigated from a computational perspective the intelligibility of several families of Boolean classifiers: decision trees, DNF formulae, decision lists, random forests, boosted trees, Boolean multilayer perceptrons, and binarized neural nets. The computational intelligibility of a family of classifiers has been evaluated as the set of XAI queries that are tractable when the classifier at hand belongs to the family. Considering a set of 9 XAI queries as a base line, we have shown the existence of a large computational intelligibility gap between the families of classifiers. Roughly speaking, the results obtained show that though decision trees are highly intelligible, the other families of classifiers we have focused on are not intelligible at all. This coheres with the commonly shared intuition that “decision trees are interpretable and other machine learning classifiers are not”, but more than that, our results give some formal ground to this intuition.

This work completes the study (Audemard, Koriche, and Marquis 2020) that focuses on designing tractable cases for a superset of the 9 XAI queries considered here, using knowledge compilation techniques. The fact that each of the 9 XAI queries is intractable (NP-hard) when the Boolean classifier $\rho$ under consideration is unconstrained justifies the need to look for specific representations of circuits into languages ensuring the tractability of those queries and for translation (“knowledge compilation”) techniques for turning classifiers into representations from such tractable languages, as it has been done in (Audemard, Koriche, and Marquis 2020).

Various perspectives of research emerge from this paper. Notably, in the present study, we have focused on representation languages which are complete for propositional logic. In other words, such representation languages are expressive enough to cover the concept class $F_n$ of all Boolean functions over $n$ variables. From the viewpoint of computational learning theory, this means that the VC dimension (Vapnik and Chervonenkis 1974) of all families of Boolean classifiers considered in this paper is equal to $2^n$. This in turn implies that these families are not efficiently PAC learnable (Valiant 1984) because their sample complexity is exponential in $n$. In fact, it is well-known that the minimal size of any Boolean multilayer neural network representing all $n$-dimensional Boolean functions must be exponential in $n$ (Shalev-Shwartz and Ben-David 2014, Theorem 20.2). So, in order to analyze the interpretability of common Boolean classifiers that can be trained using a reasonable amount of data samples, we need to focus on representation languages for which the VC dimension of the corresponding concept class is polynomial in $n$. As a prototypical example, for the class of decision lists defined over monomials of size at most $r$, the VC dimension is polynomial in $n$ when $r$ is constant; in fact, $r$-decision lists are efficiently PAC learnable (Rivest 1987). Of course, not all representation classes with polynomial VC dimension admit a polynomial-time learning algorithm, but virtually all Boolean classifiers used in practice are defined from concept classes with polynomial sample complexity. Thus, it is clear that the computational intelligibility of incomplete classes is worth being studied.

Enlarging the set of XAI queries that are used for the intelligibility assessment is another dimension for further research. Among the computational queries that could be added to the intelligibility map is the ability (or not) to compute SHAP scores in a tractable way, as investigated recently in (Arenas et al. 2021; den Broeck et al. 2021). Deriving more fine-grained complexity results (i.e., not restricted to NP-hardness in the broad sense) and determining whether the answers to some XAI queries can be approximated efficiently (under some guarantees on the quality of the approximation achieved) would be useful as well. Finally, experiments will be also needed to determine to which extent the XAI queries we have considered in the paper are hard to be addressed in practice.
Acknowledgements

Many thanks to the anonymous reviewers for their comments and insights. This work has benefited from the support of the AI Chair EXPEKCTATION (ANR-19-CHIA-0005-01) of the French National Research Agency (ANR). It was also partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

References

Anthony, M. 2001. Discrete Mathematics of Neural Networks: Selected Topics. SIAM Monographs on Discrete Mathematics and Applications.

Arenas, M.; Barceló, P.; Bertossi, L. E.; and Monet, M. 2021. The tractability of SHAP-score-based explanations for classification over deterministic and decomposable boolean circuits. In Proc. of AAAI’21, 6670–6678.

Audemard, G.; Koriche, F.; and Marquis, P. 2020. On tractable XAI queries based on compiled representations. In Proc. of KR’20, 838–849.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. Classification and Regression Trees. Wadsworth.

Breiman, L. 1996. Bagging predictors. Machine Learning 24(2):123–140.

Breiman, L. 2001. Random forests. Machine Learning 45(1):5–32.

Bunel, R.; Turkaslan, I.; Torr, P. H. S.; Kohli, P.; and Mudigonda, P. K. 2018. A unified view of piecewise linear neural network verification. In Advances in Neural Information Processing Systems 31 (NeurIPS’18), 4795–4804.

Charles, Z., and Papailiopoulos, D. 2018. Stability and generalization of learning algorithms that converge to global optima. In Proceedings of the 35th International Conference on Machine Learning (ICML’18), 745–754.

Chen, H.; Zhang, H.; Si, S.; Li, Y.; Boning, D.; and Hsieh, C. 2019. Robustness verification of tree-based models. In Advances in Neural Information Processing Systems 32 (NeurIPS’19), 12317–12328.

Crabbe, J.; Zhang, Y.; Zame, W.; and van der Schaar, M. 2020. Learning outside the black-box: The pursuit of interpretable models. In Advances in Neural Information Processing Systems (NeurIPS’20), volume 33, 17838–17849.

Darwiche, A., and Hirth, A. 2020. On the reasons behind decisions. In Proc. of ECAI’20, 712–720.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. Journal of Artificial Intelligence Research 17:229–264.

Darwiche, A., and Marquis, P. 2004. Compiling propositional weighted bases. Artificial Intelligence 157(1-2):81–113.

den Broeck, G. V.; Lykov, A.; Schleich, M.; and Suciu, D. 2021. On the tractability of SHAP explanations. In Proc of AAAI’21, 6505–6513.

Feldman, V. 2009. Hardness of approximate two-level logic minimization and PAC learning with membership queries. J. Comput. Syst. Sci. 75(1):13–26.

Flach, P. 2012. Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press.

Freund, Y., and Schapire, R. E. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In Vitányi, P. M. B., ed., Computational Learning Theory, Second European Conference, EuroCOLT ’95, Barcelona, Spain, March 13-15, 1995, Proceedings, volume 904 of Lecture Notes in Computer Science, 23–37. Springer.

Färnkrantz, J.; Gamberger, D.; and Lavrač, N. 2012. Foundations of Rule Learning. Springer.

Goodman, B., and Flaxman, S. R. 2017. European union regulations on algorithmic decision-making and a “right to explanation”. AI Magazine 38(3):50–57.

Hausssler, D. 1992. Decision theoretic generalizations of the PAC model for neural net and other learning applications. Information and computation 100(1):78–150.

Horel, E., and Giesecke, K. 2020. Significance tests for neural networks. J. Mach. Learn. Res. 21(27):1–29.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks. In Advances in Neural Information Processing Systems 29 (NeurIPS’16), 4107–4115.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019. Abduction-based explanations for machine learning models. In Proc. of AAAI’19, 1511–1519.

Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2020. On explaining decision trees. CoRR abs/2010.11034.

Jia, K., and Rinard, M. 2020. Efficient exact verification of binarized neural networks. In Advances in Neural Information Processing Systems 33 (NeurIPS’20).

Kearns, M., and Vazirani, U. 1994. An Introduction to Computational Learning Theory. MIT Press.

Koricke, F.; Lagniez, J.-M.; Marquis, P.; and Thomas, S. 2013. Knowledge compilation for model counting: Affine decision trees. In Proc. of IJCAI’13, 947–953.

Koricke, F.; Berre, D. L.; Lonca, E.; and Marquis, P. 2016. Fixed-parameter tractable optimization under DNNF constraints. In Proc. of ECAI’16, 1194–1202.

Lipton, Z. C. 2018. The mythos of model interpretability. Communications of the ACM 61(10):36–43.

Marques-Silva, J.; Gerspacher, T.; Cooper, M. C.; Ignatiev, A.; and Narodytska, N. 2020. Explaining naive bayes and other linear classifiers with polynomial time and delay. In Advances in Neural Information Processing Systems 33 (NeurIPS’20).

Narodytska, N.; Kasiviswanathan, S. P.; Ryzhyk, L.; Sagiv, M.; and Walsh, T. 2018. Verifying properties of binarized neural networks. In Proc. of AAAI’18, 6615–6624.

Narodytska, N.; Zhang, H.; Gupta, A.; and Walsh, T. 2020. In search for a sat-friendly binarized neural network architecture. In Proc. of ICLR’20.

Narodytska, N. 2018. Formal analysis of deep binarized neural networks. In Proc. of IJCAI’18, 5692–5696.
Pitt, L., and Valiant, L. 1988. Computational limitations on learning from examples. *Journal of the ACM* 35(4):965–984.

Plumb, G.; Molitor, D.; and Talwalkar, A. 2018. Model agnostic supervised local explanations. In *Advances in Neural Information Processing Systems (NeurIPS’18)*, 2520–2529.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1(1):81–106.

Ramamurthy, K.-N.; Vinzamuri, B.; Zhang, Y.; and Dhandhar, A. 2020. Model agnostic multilevel explanations. In *Advances in Neural Information Processing Systems 33 (NeurIPS’20)*.

Rivest, R. L. 1987. Learning decision lists. *Machine Learning* 2(3):229–246.

Schapire, R., and Freund, Y. 2012. *Boosting: Foundations and Algorithms*. MIT Press.

Schapire, R. E. 1990. The strength of weak learnability. *Mach. Learn.* 5:197–227.

Shalev-Shwartz, S., and Ben-David, S. 2014. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

Shalev-Shwartz, S.; Shamir, O.; Srebro, N.; and Sridharan, K. 2010. Learnability, stability and uniform convergence. *Journal of Machine Learning Research* 11(Oct):2635–2670.

Shih, A.; Choi, A.; and Darwiche, A. 2018. A symbolic approach to explaining Bayesian network classifiers. In *Proc. of IJCAI’18*, 5103–5111.

Shih, A.; Darwiche, A.; and Choi, A. 2019. Verifying binarized neural networks by Angluin-style learning. In *Proc. of SAT’19*, 354–370.

Srinivasan, A.; Vig, L.; and Bain, M. 2019. Logical explanations for deep relational machines using relevance information. *J. Mach. Learn. Res.* 20(130):1–47.

Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM* 27(11):1134–1142.

Valiant, L. 1985. Learning disjunction of conjunctions. In *Proc. of IJCAI’85*, 560–566.

Vapnik, V., and Chervonenkis, A. 1974. *Theory of Pattern Recognition*. Nauka, Moskow (in Russian).

Vapnik, V. 1998. *Statistical learning theory*. Wiley.