On Translations between ML Models for XAI Purposes

Alexis de Colnet[1] and **Pierre Marquis**[2,3]

[1] Algorithms and Complexity Group, TU Wien, Vienna, Austria
[2] Univ. Artois, CNRS, Centre de Recherche en Informatique de Lens (CRIL), F-62300 Lens, France
[3]Institut Universitaire de France
decolnet@ac.tuwien.ac.at, marquis@cril.univ-artois.fr

Abstract

In this paper, the succinctness of various ML models is studied. To be more precise, the existence of polynomial-time and polynomial-space translations between representation languages for classifiers is investigated. The languages that are considered include decision trees, random forests, several types of boosted trees, binary neural networks, Boolean multilayer perceptrons, and various logical representations of binary classifiers. We provide a complete map indicating for every pair of languages $\mathcal{C}$, $\mathcal{C}'$ whether or not a polynomial-time / polynomial-space translation exists from $\mathcal{C}$ to $\mathcal{C}'$. We also explain how to take advantage of the resulting map for XAI purposes.

1 Introduction

During the past decades, the main motivation for designing new ML models was to improve the *learning performance*. More recently, the quest for eXplainable AI (XAI) [Goodman and Flaxman, 2017; Gunning, 2019] has led to consider another goal than pure learning performance when opting for an ML model, namely *its ability to offer XAI facilities* (see e.g., [Adadi and Berrada, 2018; Miller, 2019; Samek *et al.*, 2019; Guidotti *et al.*, 2019; Srinivasan and Chander, 2020; Molnar, 2019; Lundberg *et al.*, 2020; Rudin *et al.*, 2021]). Beyond predictions, explanations of the predictions are sought for. In the same vein, verification queries enabling to check whether the predictor under consideration behaves as expected are looked for.

A major concern is that the two objectives (accuracy and XAI) can be antagonistic. Thus, predictors with high prediction performance are usually considered as poorly intelligible (see e.g., [Molnar, 2019; Arrieta *et al.*, 2020; Caruana *et al.*, 2020]), even if the lack of algorithms for answering XAI queries in practice does not always come with strong arguments against the existence of such algorithms (the absence of efficient algorithms not implying their impossibility). That mentioned, the need for explanation and verification hampers the further applications of ML machinery, especially when those applications directly affect human beings. Our society now calls for trustworthy AI, hence it is important to investigate approaches for providing ML models with XAI facilities.

In this perspective, our work falls under the so-called *formal XAI umbrella* [Marques-Silva and Ignatiev, 2022] and is relevant to *the model-specific approach to post-hoc explanation*. Thus, a predictor is supposed to have been learned and the goal is to equip it with XAI methods, allowing users to reason about the behavior of the predictor, and to understand why instances have been classified as they have been, or not classified as the user would expect. We assume that the learned predictor can be associated with a circuit that has precisely the same behaviour in terms of inputs/outputs (see e.g., [Narodytska *et al.*, 2018; Shih *et al.*, 2018; Shih *et al.*, 2019]). Then, the XAI facilities that are required can be delegated to the corresponding circuit (automated reasoning techniques are usually leveraged to achieve them) [Darwiche and Hirth, 2020; Barceló *et al.*, 2020; Parmentier and Vidal, 2021]. Using such an approach, no approximation of the predictor is performed and as a consequence, rigorous explanations [Ignatiev, 2020] can be derived (i.e., the predictor itself is explained, not a surrogate). This is of utmost importance when ML techniques are considered in safety-critical applications.

For the past few years, many XAI algorithms have been designed and evaluated for various ML models, and the collection of such algorithms is ever-increasing (see [Marques-Silva, 2022] for a recent survey). The research question considered in the present work is about the existence of *translations between ML models*. The goal is to determine whether or not one can take advantage of XAI algorithms when dealing with ML models other than that for which they have been conceived, provided that a sufficiently efficient translation between the models exists. Addressing this issue amounts to determining *how computationally demanding it is to translate a predictor from an input ML model into a predictor from another (output) model for which XAI facilities are available*. In this paper, we are interested in equivalence-preserving translations (i.e., the output predictor is equivalent to the input predictor) since we want to guarantee that the explanations computed when rigorous for the output predictor are also rigorous for the input predictor.

To make things concrete, suppose that a predictor $f_{\mathcal{C}}$ from a given ML model $\mathcal{C}$ (e.g., $\mathcal{C}$ is the class BT of boosted trees) has been learned and that we want to derive contrastive explanations (see e.g., [Miller, 2019; Guidotti, 2022]) for instances x classified by $f_{\mathcal{C}}$, but we do not have an XAI algorithm to

perform this task. Suppose that a translation τ exists from $\mathcal{C}$ to another ML model $\mathcal{C}'$ (e.g., $\mathcal{C}'$ is the class BMP of Boolean multilayer perceptrons), and that an XAI algorithm xai (e.g., to derive a contrastive explanation for an instance $\boldsymbol{x}$) suited to any predictor $f_{\mathcal{C}'}$ from $\mathcal{C}'$ is available. Then we immediately get an XAI algorithm for the same task but now suited to any predictor $f_{\mathcal{C}}$ from $\mathcal{C}$: just compute $xai(\tau(f_{\mathcal{C}}), \boldsymbol{x})$.

Computationally speaking, we are interested in two types of translation: *polynomial-time translations*, and *polynomial-space* translations. A polynomial-time translation from an ML model $\mathcal{C}$ to an ML model $\mathcal{C}'$, noted $\mathcal{C} \leq_p \mathcal{C}'$, ensures that any polynomial-time XAI algorithm for $\mathcal{C}'$ can be turned into a polynomial-time XAI algorithm for $\mathcal{C}$. Polynomial-space translations, noted $\mathcal{C} \leq_s \mathcal{C}'$, guarantee that the output predictor from $\mathcal{C}$ is of size polynomial in the size of the input predictor from $\mathcal{C}'$. Clearly, if $\mathcal{C} \leq_p \mathcal{C}'$ holds then $\mathcal{C} \leq_s \mathcal{C}'$ holds as well, but the converse is false in general. Indeed, no assumptions are made on the time needed to produce the output predictor when $\mathcal{C} \leq_s \mathcal{C}'$ holds. Polynomial-space translations can be nevertheless useful when no XAI algorithms are available for $\mathcal{C}$, or when the computation time required for achieving the translation can be balanced by considering sufficiently many queries (and explanation queries can indeed be numerous since the instances to be explained are numerous as well). When no polynomial-space translation exists from $\mathcal{C}'$ to $\mathcal{C}$, there is no way to turn the polynomial-time XAI algorithms for $\mathcal{C}'$ into polynomial-time XAI algorithms for $\mathcal{C}$.

In this work, we initiate a systematic comparison of ML models, viewed as classes of binary classifiers, with respect to the $\leq_p$ and the $\leq_s$ relationships. Thus, for the classifiers considered, input variables are over $\{0, 1\}$ and a classifier maps every assignment to its variables to either 0 or 1. We consider well-known models of binary classifiers, that are conveniently summarized together in [Audemard *et al.*, 2021]. These models are briefly described in this paragraph and in more details in Section 2. Our contributions are spread in the paper as follows. In Section 3, we consider tree-based classifiers that are linear predictors over *decision trees* (DT). These are sets of decision trees $T_1, \ldots, T_m$ such that an assignment is mapped to 1 if and only if the sum of the outputs of the decision trees exceeds a certain threshold. We compare *random forests* (RF): every tree can return $+1$ or -1; *boosted trees* with trees T_i associated with weights $w_i \in \mathbb{Q}$ (BT): every tree T_i can return $+w_i$ or $-w_i$; and *boosted regression trees* (BRT), where weights are associated with tree leaves, and every tree can return any number in $\mathbb{Q}$. We show a strong separation between RF and BT (and BRT), in the sense that BT $\leq_p$ RF but RF $\not\leq_s$ BT. We also describe a, perhaps unexpected, polynomial-time translation from BRT to BT. In Section 4, we look at network models, with *binary neural networks* (BNN) and *Boolean multilayer perceptrons* (BMP). There, we show a close connection between these models and threshold circuits using "small" integer weights ($\widehat{\text{TH}}$) and threshold circuits using any integer weights (TH). We lean on this connection to strongly separate BNN and BMP from tree-based classifiers, and to show that BNN $\leq_p$ BMP. In Section 5, we add to the analysis a few other classes of representations of Boolean functions (which can be seen as classes of binary classifiers). Those classes include the well-
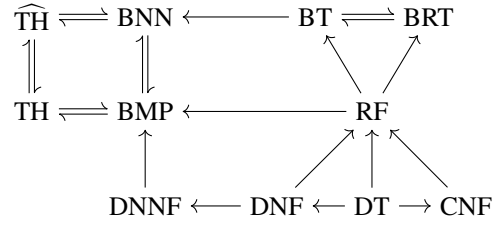

Figure 1: The $\leq_p, \leq_s$-map for different classes of binary classifiers.

known formulas in conjunctive normal form (CNF) and in disjunctive normal form (DNF), and the (less well-known) circuits in decomposable negation normal form (DNNF). These circuits have recently been suggested as useful representations for answering XAI queries [Audemard *et al.*, 2020; Darwiche and Hirth, 2020]. Other related work are discussed in Section 6.

Our results are summarized in Figure 1, presenting a Hasse diagram that can be read as follows for *both* $\leq \, \in \{\leq_p, \leq_s\}$:

- an arrow $\mathcal{C} \to \mathcal{C}'$ means that $\mathcal{C}' \leq \mathcal{C}$ but that $\mathcal{C} \not\leq \mathcal{C}'$,
- a double harpoon $\mathcal{C} \rightleftharpoons \mathcal{C}'$ means that $\mathcal{C} \leq \mathcal{C}'$ and $\mathcal{C}' \leq \mathcal{C}$,
- no link between $\mathcal{C}$ and $\mathcal{C}'$ means either that the relationship can be derived by transitivity, or that $\mathcal{C} \not\leq \mathcal{C}'$ and $\mathcal{C}' \not\leq \mathcal{C}$.

Notably, most polynomial-time translations pointed out in the following can be envisioned in practice since they run in *linear time* in the size of the initial classifier. A notable exception is the translation from BMP to BNN, that runs in time at least quadratic in the size of the classifier. Our results are discussed in Section 7. Due to space constraints, many proofs are deferred to the full version of this paper available here: www.cril.fr/expekctation/papers/deColnet-Marquis-IJCAI23-full-proof.pdf.

2 Preliminaries

A *Boolean variable* is a variable over $\{0, 1\}$. A *literal* is a Boolean variable x or its negation $\overline{x}$. An *assignment* a to a set X of Boolean variables is a mapping from X to $\{0, 1\}$. We denote by $\{0, 1\}^X$ the set of all assignments to X. A *Boolean function* over X is a mapping from $\{0, 1\}^X$ to $\{0, 1\}$. The symbols $\land, \lor$ are used to denote conjunction and disjunction, respectively. A *clause* is a disjunction of literals and a *term* is a conjunction of literals. A formula in CNF is a conjunction of clauses and a formula in DNF is a disjunction of terms.

2.1 Tree-Based Models

Let X be a finite set of Boolean variables and let a be an assignment to X. A *decision node* labelled by $x \in X$ is a node of the form ⟨x⟩ interpreted as follows: if x is set to 0 (resp. 1) then follow the dashed (resp. plain) arrow.

Decision tree. A *decision tree* over X and about a set Y [Breiman *et al.*, 1984; Quinlan, 1986] represents a mapping from $\{0, 1\}^X$ to Y. It is a rooted tree where internal nodes are decision nodes labelled by variables in X, and leaves are elements in Y. We assume wlog that any variable labels at most one node of every root-to-leaf path. DT_Y denotes the

set of decision trees over X about Y. For convenience we write $\mathrm{DT} = \mathrm{DT}_{\{0,1\}}$ and $\mathrm{DT}_{\pm} = \mathrm{DT}_{\{-1,1\}}$. The label of the leaf obtained when following the path consistent with a complete assignment a in the tree T is denoted by $T(a)$.

Random forest. Decision trees 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. Thus, a *random forest* over X [Breiman, 2001] is a finite set of binary decision trees over X in $\mathrm{DT}_{\pm}$. The random forest classifier composed of the trees $T_1, T_2, \ldots, T_m$ represents the Boolean function over X that evaluates to 1 on a if and only if $\sum_{i=1}^m T_i(a) \geq 0$. RF denotes the class of random forests.

Boosted tree. Tree ensembles can also be trained using the boosting technique. Thus, a *boosted tree* is a random forest whose trees T_i or tree leaves are associated with weights w_i in $\mathbb{Q}$. When weights are associated with trees, the classifier corresponding to $(T_1, w_1), \ldots, (T_m, w_m)$ represents the Boolean function that evaluates to 1 on a if and only if $\sum_{i=1}^m w_i \cdot T_i(a) \geq 0$. When the weights are not associated with the trees but with their leaves, we end up with a forest whose trees are from $\mathrm{DT}_{\mathbb{Q}}$ (equivalently, one can view them as regression trees). The classifier corresponding to $T_1, \ldots, T_m \in \mathrm{DT}_{\mathbb{Q}}$ represents the Boolean function that evaluates to 1 on a if and only if $\sum_{i=1}^m T_i(a) \geq 0$. BT denotes the class of boosted trees with trees associated with weights, and BRT the class of boosted trees with tree leaves associated with weights (aka "boosted regression trees").

Standard learning algorithms for generating boosted trees output predictors from BT (e.g., this is the case of AdaBoost [Freund and Schapire, 1997; Schapire and Freund, 2014]) or they output predictors from BRT (e.g., this is the case of XGBoost [Chen and Guestrin, 2016], LightGBM [Ke *et al.*, 2017] and CatBoost [Prokhorenkova *et al.*, 2018]).

We denote by $|T|$ the number of nodes (including leaves) of the decision tree T. More generally, the *size* of a classifier C is written $|C|$. If C is a random forest $T_1, \ldots, T_m$, then $|C| = \sum_{i=1}^m |T_i|$. If C is a boosted tree $T_1, \ldots, T_m$, then $|C|$ is $\sum_{i=1}^m |T_i|$ plus the space needed to store the weights in memory. We will often assume that the weights are in $\mathbb{Z}$ and are encoded in binary with no limit on the number of bits (note by the way that $|w_i|$ denotes the absolute value of w_i and not the number of bits in its encoding).

2.2 Network Classifiers

Boolean multilayer perceptrons. A (feedforward) neural network is a directed acyclic graph whose edges are labeled with weights in $\mathbb{Q}$, and nodes (alias neurons) correspond to input variables over $\mathbb{Q}$ when their fan-in is 0, and to functions over their weighted inputs otherwise. In a *multilayer* neural network, the nodes are partitioned in sets $L_1, \ldots, L_d$ called layers such that any edge is directed from a node of L_i to a node of L_{i+1} for some $i < d$. The *input layer* L_1 contains only the input variables. The nodes of the *output layer* L_d all have fan-out 0 and compute the output vector of the network. For every $i < d$, the layers L_i and L_{i+1} are supposed *fully connected* in the sense that all nodes of L_i are connected to all nodes of L_{i+1}. A *Boolean multilayer perceptron* (see

e.g., [Anthony, 2001]) is a multilayer neural network whose input variables are Boolean, whose output layer contains a single node, and whose non-input nodes are threshold gates. Formally, a *threshold gate* over inputs $x_1, \ldots, x_n$, associated respectively to rational numbers (weights) $w_1, \ldots, w_n$, is a function

$$\mathrm{sgn}\left(\sum_{i=1}^n w_i \cdot x_i - \theta \right)$$

where $\mathrm{sgn}(k) = -1$ if $k < 0$ and $\mathrm{sgn}(k) = 1$ otherwise, and θ is a fixed number in $\mathbb{Q}$. If the inputs are variables over $\{0, 1\}$, then the function is equivalent to $\mathrm{sgn}(\sum_{i=1}^n w_i' \cdot x_i' - \theta')$ where $w_i' = -w_i$, $\theta' = 2\theta - \sum_{i=1}^n w_i$ and $x_i' = 1 - 2x_i$ takes value in $\{-1, 1\}$. So we will often assume that the inputs of the circuits are over $\{-1, 1\}$. We denote by BMP the class of all Boolean multilayer perceptrons. We view BMP as a class of representations for Boolean functions: we just have to assume that the output gate returns 0 or 1 instead of 1 or -1 (respectively). [1]

Binarized neural network. *Binarized neural networks (BNN)* [Hubara *et al.*, 2016] can be seen as Boolean multilayer perceptrons whose weights range in $\{-1, 1\}$, that is, all threshold gates are of the form $y_j = \mathrm{sgn}(\sum_{i=1}^n s_{i,j} \cdot x_i - \theta_j)$ where $s_{i,j} \in \{-1, 1\}$. For completeness, we nevertheless consider as well the following more technical but more standard description. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_m)$ be respectively the input vector and the output vector of a layer L where all variables x_i and y_i ranges in $\{-1, 1\}$. In a BNN, it is usual to break the threshold gates into three parts, in particular y_j is computed as follows, where $b_j, \alpha_j, \mu_j, \nu_j, \gamma_j \in \mathbb{Q}$ and $\nu_j \neq 0$:

1. Linear transformation: compute $y_j' = \sum_{i=1}^n s_{i,j} \cdot x_i + b_j$

2. Batch normalization: compute $y_j'' = \alpha_j \frac{(y_j' - \mu_j)}{\nu_j} + \gamma_j$

3. Binarization: $y_j = \mathrm{sgn}(y_j'')$.

Assuming $\alpha_j \neq 0$, one indeed has $y_j = \mathrm{sgn}(\sum_{i=1}^n s_{i,j} \cdot x_i - \theta_j)$ where $\theta_j = \mu_j - b_j - \nu_j \gamma_j / \alpha_j$ (if $\alpha_j = 0$, then y_j is a constant and the gate can be removed from L while modifying parameters of the next layer). It will be simpler to consider that the nodes of a BNN are threshold gates with weights in $\{-1, 1\}$ than to use the three-step computation. Just as for Boolean multilayer perceptrons, we assume whenever it is convenient that the inputs of the circuits are variables in $\{-1, 1\}$ and not variables in $\{0, 1\}$. We denote by BNN the class of all binarized neural networks, which we again view as a class of representations for Boolean functions.

The size of a BMP or of a BNN classifier C is its number of edges, plus the number of bits needed to write all the weights and thresholds used in threshold gates.

2.3 Translations

A class $\mathcal{C}$ of representations for Boolean functions is *fully expressive* when every Boolean function over finitely many variables has a representation in $\mathcal{C}$. In the following, the classes $\mathcal{C}$ and $\mathcal{C}'$ are fully expressive.

[1] An important research strand on Boolean functions considers variables over $\{-1, 1\}$ instead of $\{0, 1\}$. The usual translation is "1 becomes 0, and -1 becomes 1", see for instance [O'Donnell, 2014].

Polynomial-time translations. We say that there is a *polynomial-time translation* from $\mathcal{C}'$ to $\mathcal{C}$, denoted by $\mathcal{C} \leq_p \mathcal{C}'$, when there is a polynomial-time algorithm that, for every $C' \in \mathcal{C}'$ returns some $C \in \mathcal{C}$ representing the same function as C'. When $\mathcal{C} \leq_p \mathcal{C}'$ and $\mathcal{C}' \leq_p \mathcal{C}$ hold, we note $\mathcal{C} \simeq_p \mathcal{C}'$.

Polynomial-space translations. We say that there is a *polynomial-space translation* from $\mathcal{C}'$ to $\mathcal{C}$ (or, equivalently, that $\mathcal{C}$ is *at least as succinct as* $\mathcal{C}'$), denoted by $\mathcal{C} \leq_s \mathcal{C}'$, when there is a polynomial p such that, for every $C' \in \mathcal{C}'$, there exists a representation $C \in \mathcal{C}$ of the same function whose size is $|C| \leq p(|C'|)$. We say that $\mathcal{C}$ is *strictly more succinct than* $\mathcal{C}'$ when $\mathcal{C} \leq_s \mathcal{C}'$ but $\mathcal{C}' \not\leq_s \mathcal{C}$. We say that $\mathcal{C}$ and $\mathcal{C}'$ are *equally succinct*, denoted by $\mathcal{C} \simeq_s \mathcal{C}'$, when $\mathcal{C} \leq_s \mathcal{C}'$ and $\mathcal{C}' \leq_s \mathcal{C}$.

Clearly enough, $\leq_p$ and $\leq_s$ are *reflexive* and *transitive*. Furthermore, $\mathcal{C}' \subseteq \mathcal{C}$ implies $\mathcal{C} \leq_p \mathcal{C}'$ and $\mathcal{C} \leq_p \mathcal{C}'$ implies $\mathcal{C} \leq_s \mathcal{C}'$.

3 Translations between Tree-Based Models

In this section, we describe the $\leq_p$ and $\leq_s$ relationships between the classes of binary classifiers DT, RF, BT, BRT. For most pairs of classes, one side of the relationship is trivial. In particular, it is clear that BRT $\leq_p$ BT $\leq_p$ RF $\leq_p$ DT. It remains to study the other direction. Using a function as simple as $majority_n$ over n variables that evaluates to 1 if and only if at least $\lfloor n/2 \rfloor$ variables are set to 1, we can separate RF from DT in terms of succinctness:

Proposition 1. *We have* RF $\leq_p$ DT *and* DT $\not\leq_s$ RF.

3.1 Comparing BT and RF

Weights used in boosted trees can, in theory, be arbitrary numbers in $\mathbb{Q}$, but they can all be multiplied by the least common multiple of their denominators without modifying the function represented by the classifier. For reasonable representations of rationals, this only results in a polynomial-size increase on the space needed to store the weights. So, from now on, we assume that all weights are in $\mathbb{Z} \setminus \{0\}$.

When the weights $w_1, \ldots, w_n$ are integers, $w_1 \cdot T_1 + \cdots + w_m \cdot T_m$ can be turned into an *unweighted* sum of decision trees by creating, for every $i \leq m$, $|w_i|$ clones $T_i^1, T_i^2, \ldots, T_i^{|w_i|}$ of T_i where the sign of a leaf is switched when $w_i < 0$. Indeed, we have that $w_i \cdot T_i = \sum_{j=1}^{|w_i|} T_i^j$ so the boosted tree $(T_1, w_1), \ldots, (T_m, w_m)$ is equivalent to the random forest $\bigcup_{i=1}^m \{T_i^1, \ldots, T_i^{|w_i|}\}$. Clearly this translation is useful only when $max_i |w_i|$ is small, but it teaches us that the encoding of integer weights in memory is important. In particular, if the weights were encoded in unary, or if they were encoded in binary with a fixed number of bits, then BT and RF would be equally succinct, even though the translation from BT to RF is impractical. But because we assume that weights are represented in binary with no limit on the number of bits, it turns out that RF $\not\leq_s$ BT. To see this, we lean on a result of Goldmann, Håstad and Razborov [Goldmann *et al.*, 1992]. Consider the Boolean variables $X \cup Y := \{x_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq 4n\} \cup \{y_j \mid 1 \leq j \leq 2n\}$ and let:

$$p(X,Y) = \begin{cases} 0 & \text{if } 1 + \sum_{i,j} 2^i y_j'(x_{i,2j}' + x_{i,2j+1}') \geq 0 \\ 1 & \text{otherwise} \end{cases}$$

where $x' = 1 - 2x$. Goldmann et al. show that $p(X,Y)$ is hard to represent with depth-2 threshold circuits (a circuit that has two levels of threshold gates) using small weights.

Proposition 2. [Goldmann *et al.*, 1992, Corollary 7] *If $p(X,Y)$ is computed by a depth-2 threshold circuit with weights bounded by w and size s then $sw^2 \geq \Omega(\frac{2^{n/2}}{n^{5/2}})$.*

We use $p(X,Y)$ to separate BT and RF in terms of succinctness. First, it is easy to represent $y_j' x_{i,2j}'$ and $y_j' x_{i,2j+1}'$ by decision trees, so it is easy to represent $p(X,Y)$ in BT.

Proposition 3. *There exists a set of boosted trees of size $O(n^2)$ that represents $p(X,Y)$.*

But we can also show that, if there was a small random forest representing $p(X,Y)$, then there would be a small depth-2 threshold circuit computing $p(X,Y)$ using only small weights which, by Proposition 2, cannot be.

Proposition 4. *Every random forest representing $p(X,Y)$ has size $\Omega(\frac{2^{n/2}}{n^{5/2}})$.*

Combining Proposition 3 and Proposition 4, we obtain the following.

Proposition 5. *We have* BT $\leq_p$ RF *and* RF $\not\leq_s$ BT.

3.2 Comparing BRT and BT

We now give a polynomial-time translation from boosted regression trees to boosted trees.

Proposition 6. *There is a quadratic-time translation from BRT to BT. Thus* BT $\leq_p$ BRT.

Proof. Let $T_1, \ldots, T_m$ be the trees composing $C \in$ BRT. Recall that their leaves are labelled by integers. Let W_i be the set of integers distinct from 0 that label leaves of T_i. Note that $|W_i| \leq |T_i|$. For every $w \in W_i$, let $T_{i,w}$ be the decision tree obtained from T_i by changing to 0 the weight of every leaf that is not w. Observe that $T_i = \sum_{w \in W_i} T_{i,w}$. Next let $T_{i,w}^-$ be the decision tree obtained from $T_{i,w}$ by replacing every weight 0 by -1 and by replacing every weight w by 1, and let $T_{i,w}^+$ be a decision tree that consists of a single leaf labelled by 1. Note that $T_{i,w}^-$ and $T_{i,w}^+$ are in $\mathsf{DT}_\pm$ and that $T_{i,w} = \frac{w}{2}(T_{i,w}^- + T_{i,w}^+)$. Thus $\sum_{i=1}^m T_i \geq 0$ is equivalent to

$$\sum_{i=1}^m \sum_{w \in W_i} T_{i,w} \geq 0 \quad \Leftrightarrow \quad \sum_{i=1}^m \sum_{w \in W_i} \frac{w}{2}(T_{i,w}^- + T_{i,w}^+) \geq 0$$

$$\Leftrightarrow \quad \sum_{i=1}^m \sum_{w \in W_i} w T_{i,w}^- + \left(\sum_{i=1}^m \sum_{w \in W_i} w\right) T_0 \geq 0$$

where T_0 is the tree consisting of a single leaf labelled by 1. The last inequality corresponds to a classifier C' in BT composed of $1 + \sum_{i=1}^m |W_i|$ trees. For every T_i with $i \geq 1$ and every $w \in W_i$, $T_{i,w}^-$ is constructed in time $O(|T_i|)$. So constructing the trees in C' takes time $O(\sum_i (|W_i| \times |T_i|)) \leq O(\sum_i |T_i|^2)$. Finally, the weights of C' are exactly those of C except for the weight of T_0 which is a sum of integers computed in time linear in the number of bits needed to write all weights. $\square$

$$= \quad 2 \times (\overline{x}_1 \wedge x_2) \quad + \quad 4 \times (\overline{x}_1 \wedge \overline{x}_3) \quad + \quad 3 \times (\overline{x}_1 \wedge \overline{x}_2) \quad + \quad 3 \times (x_1 \wedge x_3)$$
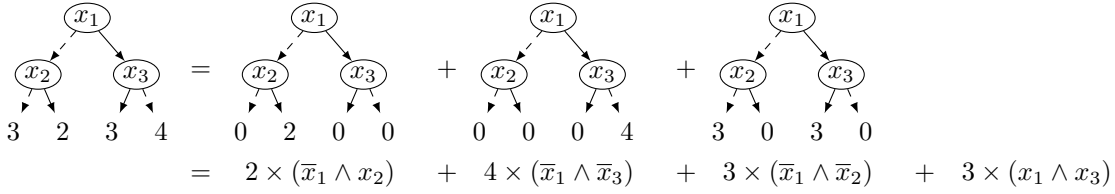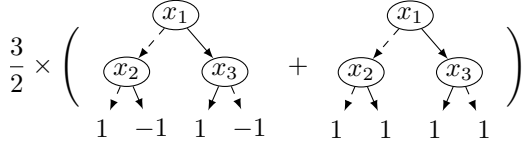
Figure 2: Decomposition of a decision tree as a weighted sum of terms

To illustrate the translation used in the proof above, consider the tree T_i in Figure 2 and its decomposition. The first line corresponds to the decomposition $T_i = T_{i,2} + T_{i,4} + T_{i,3}$. We have that $T_{i,3}$ is equivalent to

which corresponds to the decomposition $T_{i,3} = \frac{3}{2} \times (T_{i,3}^- + T_{i,3}^+)$ where $T_{i,3}^+$ is equivalent to 1.

4 Translations between Network Classifiers

In this section, we describe a close connection between BMP, BNN and the following classes of threshold circuits over X, i.e., Boolean circuits where internal nodes consist only of threshold gates:

- TH: the class of threshold circuits whose weights are integers.

- $\widehat{\text{TH}}$: the subclass of TH using only weights whose absolute value is at most polynomial in the number of variables.

- $\widehat{\text{TH}}_\pm$: the subclass of $\widehat{\text{TH}}$ using only weights 1 and -1.

It is folklore that $\widehat{\text{TH}}$ and $\widehat{\text{TH}}_\pm$ are equally succinct. Indeed, on the one hand, $\widehat{\text{TH}}_\pm \subset \widehat{\text{TH}}$ so $\widehat{\text{TH}} \leq_p \widehat{\text{TH}}_\pm$. On the other hand, for every edge labelled by $w > 0$ connecting the output of a gate g to the input of a gate p in a circuit $C \in \widehat{\text{TH}}$, one can remove the edge and insert $|w|$ dummy gates between g and p as in Figure 3; the resulting circuit is in $\widehat{\text{TH}}_\pm$ and its number of gates is at most $\max_w |w|$ times the number of gates in C. This translation can be done in linear time in the sum of weights in C, so $\widehat{\text{TH}}_\pm \leq_p \widehat{\text{TH}}$. More surprisingly perhaps, every threshold circuit of depth d can be simulated by a threshold circuit of depth $2d$ in $\widehat{\text{TH}}$ with only a polynomial size increase. We owe this result to Goldmann, Håstad and Razborov [Goldmann *et al.*, 1992]. It follows that TH, $\widehat{\text{TH}}$ and $\widehat{\text{TH}}_\pm$ are equally succinct.

4.1 BMP and Threshold Circuits

By definition, a Boolean multilayer perceptron is a threshold circuit whose particularity is to be *layered*, that is, its gates are partitioned into sets $L_1, \ldots, L_d$ such that, for every $1 \leq k \leq d-1$, the inputs of every gate in L_{k+1} are only connected to the output of gates in L_k. Recall that the layers are *fully*
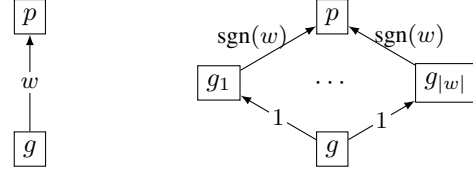

Figure 3: Introducing clone gates to reduce the weights.

connected. It is readily verified that any threshold circuit C can be turned in polynomial time into an equivalent threshold circuit that is layered and fully connected: denoting by d the depth of C and by e its number of edges, we add at most $d \times e$ gates computing functions the form $y = \text{sgn}(x)$ to make C layered, then we add edges with weight 0 to fully connect successive layers. Thus, the following holds:

Proposition 7. *We have* BMP $\simeq_p$ TH.

Interestingly, polynomial-time translations exist also between BMP and $\widehat{\text{TH}}_\pm$ due to the following:

Proposition 8. *We have* TH $\simeq_p \widehat{\text{TH}} \simeq_p \widehat{\text{TH}}_\pm$.

The main difficulty in the proof of Proposition 8 is the relationship $\widehat{\text{TH}} \leq_p$ TH. Indeed, the proof that $\widehat{\text{TH}} \leq_s$ TH by Goldmann et al. does not directly extend from $\leq_s$ to $\leq_p$, partly because it involves probabilistic arguments. We have found a polynomial-time translation from TH to $\widehat{\text{TH}}$ that, unfortunately, does not guarantee that the depths of circuits before and after translation stay within a constant factor (as in the proof of $\widehat{\text{TH}} \leq_s$ TH). Thus, our polynomial-time translation may be impractical. Improvements are left for future work, in this paper we only prove the existence of the polynomial-time translation.

4.2 BNN and Small-Weights Threshold Circuits

Consider an internal layer of a BNN whose input is a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ over $\{-1, 1\}$ and whose output is a vector of size $\boldsymbol{y} = (y_1, \ldots, y_m)$ over $\{-1, 1\}$. Recall that the computation of y_j reduces to a single threshold function $\text{sgn}(\sum_{i=1}^n s_{i,j} x_i - \theta_j)$ for some $s_{i,j} \in \{-1, 1\}$ and some integer θ_j. So a BNN is a particular threshold circuit that is layered, that is fully connected, and that uses only weights $+1$ and -1. Now, we show the following proposition:

Proposition 9. *There is a polynomial-time procedure that, given a circuit in* $\widehat{\text{TH}}_\pm$ *of depth d, returns an equivalent BNN composed of at most $d + 1$ layers. Hence* BNN $\simeq_p \widehat{\text{TH}}_\pm$.

We need several intermediate results to prove Proposition 9. First, we can easily render a threshold circuit in $\widehat{\text{TH}}_\pm$ layered
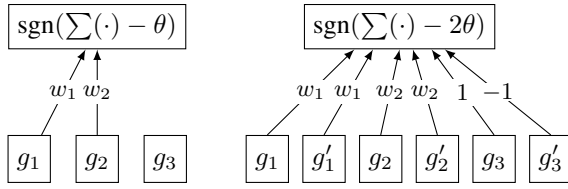
Figure 4: Clone gates to fully connect layers.

in polynomial time using the procedure previously described for circuits in TH. So, we have the following proposition:

Proposition 10. *Every threshold circuit $C \in \widehat{\mathrm{TH}}_{\pm}$ can be transformed in polynomial time into an equivalent layered threshold circuit $C' \in \widehat{\mathrm{TH}}_{\pm}$.*

But then it remains to make the layers fully connected, a task which, since weights 0 are not allowed, requires a bit more work. Consider a layer L_k with $k > 1$ that is not the last layer (so L_{k+1} exists). For every gate $g \in L_k$, we add a clone g' of g to L_k. We call g an *original* gate and g' its *clone* gate. We ensure that $g' \equiv g$ by connecting g' to L_{k-1} exactly as g. For every gate $p \in L_{k+1}$ and every *original* $g \in L_k$, if g is already connected to p with weight w, then we connect g' to p with the same weight. If g is not connected to p, then we connect both g and its clone g' to p with opposite weights. Since $g \equiv g'$, gates that were not originally connected to p still contribute 0 to p, while those that were already connected to p now contribute twice their weight. So it only remains to double the value of the threshold of p to preserve the function it computes. This simple trick, illustrated in Figure 4, allows us to fully connect every layer L_k to L_{k+1} for $k > 1$, in exchange for doubling the size of the layers. This leads to the following proposition:

Proposition 11. *Every layered circuit $C \in \widehat{\mathrm{TH}}_{\pm}$ whose first layer is fully connected to the second can be transformed in polynomial time into an equivalent layered circuit $C' \in \widehat{\mathrm{TH}}_{\pm}$ whose layers are all fully connected.*

We cannot proceed as before to fully connect L_1 to L_2 since cloning input variables is forbidden. So, before using Proposition 11, we insert a new layer between L_1 and L_2, that is fully connected to L_1 and does not modify the function represented. The construction can be found in the full version of the paper.

Proposition 12. *Every circuit $C \in \widehat{\mathrm{TM}}_{\pm}$ containing k layers can be transformed in polynomial time into an equivalent circuit $C' \in \widehat{\mathrm{TM}}_{\pm}$ that contains $k + 1$ layers and whose first layer is fully connected to the second.*

Proposition 9 directly follows from Propositions 10, 11 and 12. A first consequence of Proposition 9 is that a polynomial-time translation exists between BNN and BMP.

Proposition 13. *We have BNN $\simeq_p$ BMP.*

A second consequence is that a polynomial-time translation exists between BT and BNN or BMP. Indeed, it is easy to transform a decision tree into a Boolean circuit using $\vee$-gates and $\wedge$-gates which, in turn, using that $(a \vee b) \equiv (a + b \geq 1)$ and $(a \wedge b) \equiv (a + b \geq 2)$, is transformed in linear time into

a threshold circuit. So, to construct a threshold circuit equivalent to a boosted tree classifier, it suffices to put a threshold gate using the weights of the trees on top of the threshold circuits representing the trees.

Proposition 14. *We have BMP $\leq_p$ BT and BNN $\leq_p$ BT.*

However, we also have that BMP and BNN are exponentially more succinct than BT. To prove this, we use the function $parity_n$ over n variables that evaluates to 1 if and only if the number of variables set to 1 is odd. On the one hand, there is a linear-time translation between BT and threshold functions over terms (i.e., conjunctions of literals): it suffices to rewrite every decision tree as a weighted sum of terms as shown in Figure 2. From this translation and the following result of Goldmann, one can prove that $parity_n$ is hard to represent in BT.

Proposition 15. [Goldmann, 1997, Theorem 5] *Any circuit computing $parity_n$ that is a threshold gate over $\wedge$-gates contains at least $\frac{1}{4}\left(\frac{4}{3}\right)^{n/2}$ $\wedge$-gates for n sufficiently large.*

Proposition 16. *Every classifier in BT representing $parity_n$ has size $2^{\Omega(n)}$.*

On the other hand, $parity_n$ can be computed by a threshold circuit using only two layers of $O(n)$ gates and unit weights (see for instance [Jukna, 2014, Exercise 11.10]). So we have:

Proposition 17. *We have BT $\not\leq_s$ BMP and BT $\not\leq_s$ BNN.*

5 Translations w.r.t. Compilation Languages

The existence of (polynomial-time or polynomial-space) translations between classes of representations is one of the key concerns of *knowledge compilation* [Darwiche and Marquis, 2002], a research line developed for more than 30 years in AI. The significance of compilation languages pushed the community to confront them to XAI queries (this is discussed in Section 6). Thus, in order to complete the picture, we briefly discuss the feasibility of translations connecting compilation languages to ML models. One of the most important compilation language is the class DNNF of circuits in *decomposable negation normal form*. We omit the definition of circuits in DNNF (see [Darwiche, 2001]), and only recall that DNNF is strictly more succinct than many other compilation languages [Darwiche and Marquis, 2002; Bova *et al.*, 2016]. In particular, it is known that DNNF $\leq_p$ DNF, DNF $\not\leq_s$ DNNF, CNF $\not\leq_s$ DNNF and DNNF $\not\leq_s$ CNF [Darwiche and Marquis, 2002; Bova *et al.*, 2014].

A first separation follows from Proposition 16 and the fact that $parity_n$ admits representations of size $O(n)$ in DNNF.

Proposition 18. *We have that BT $\not\leq_s$ DNNF.*

The separation holds in the other direction as well using a trick to turn any formula in CNF (or DNF) into an equivalent random forest in polynomial time (see [Audemard *et al.*, 2022, Proposition 2]). Thus, we have DNNF $\not\leq_s$ RF.

Finally, for completeness, we can show that RF is exponentially more succinct that the classes of formulas in CNF and DNF by considering the function $majority_n$ (see Proposition 2 in [Audemard *et al.*, 2022]). This function is trivial to represent with a random forest, but requires exponential

size formulas in CNF or DNF (see [Håstad *et al.*, 1995]). As a consequence, we get that CNF $\not\leq_s$ RF and DNF $\not\leq_s$ RF. When the forest is a single tree however, it is known that CNF $\leq_p$ DT, DNF $\leq_p$ DT, DT $\not\leq_s$ CNF and DT $\not\leq_s$ DNF.

6 Other Related Work

In order to achieve useful accuracy/explainability trade-offs when designing ML models, many strategies have been envisioned so far. *Explanation by design* [Du *et al.*, 2020] is about learning predictors that are considered as intrinsically interpretable, like decision trees or linear models. The key limitation of this approach is that the predictors that are considered are typically not very accurate. In *model-agnostic approaches* to XAI, the predictor under consideration is associated with a proxy from another ML model, which is considered more explainable in essence. Very popular approaches to XAI including LIME [Ribeiro *et al.*, 2016], SHAP [Lundberg and Lee, 2017], Anchors [Ribeiro *et al.*, 2018], and LORE [Guidotti *et al.*, 2019] are relevant to this research line and approximate the decision surface of a model using an explainable one. The main drawback of such model-agnostic approaches (compared to model-specific ones) is that the explanations they furnish are not rigorous: one can find "counterexamples" for them, i.e., pairs of instances that share the same explanation but are nevertheless classified differently by the predictor [Ignatiev *et al.*, 2019]. Besides, the amount of "counterexamples" can be high when the approaches listed above are used [Ignatiev, 2020].

Leveraging knowledge compilation techniques for XAI purposes has been considered so far in a couple of papers. [Audemard *et al.*, 2020] list a number of XAI queries for multi-label classifiers. On such a basis, the XAI level of an ML model can be represented as the subset of the XAI queries for which polynomial-time algorithms exist (this tractability assumption is standard in AI to separate tasks that are likely to be feasible in practice to those that are not). When such an algorithm exists, the ML model is said to offer the XAI query. [Audemard *et al.*, 2020] and [Huang *et al.*, 2022] present sufficient conditions on the representation language used to encode the predictor that ensure that XAI queries are offered, and they identify a number of compilation languages for which such conditions are satisfied (thus, the predictor can be compiled into them). [Shi *et al.*, 2020] consider the compilation of the decision function of binary neural networks into compiled representations from OBDD [Bryant, 1986] or SDD [Darwiche, 2011] (two subsets of DNNF). Among other things, the authors show how to compute the expected robustness of a neural network (a specific verification query), given an OBDD/SDD representation of it. [Darwiche and Hirth, 2020; Darwiche and Marquis, 2021; Darwiche and Ji, 2022] show how answering various XAI queries can be facilitated when the classifier under consideration has been turned into compiled forms, especially from Decision-DNNF [Huang and Darwiche, 2005] or from SDD (Decision-DNNF is another subset of DNNF). In [Audemard *et al.*, 2021], the XAI level (or "computational intelligibility") of several families of ML models (including decision trees, random forests, decision lists, multi-layer Boolean percep-

trons, binary neural networks) has been evaluated. Roughly, this paper has shown that none of the ML models considered in this evaluation, but decision trees, satisfies any XAI queries among those identified; on the opposite, every such query is offered by the decision tree model. [de Colnet and Marquis, 2022] investigates the complexity of enumerating sufficient reasons from representations in Decision-DNNF. However, the issue of determining the existence of sufficiently efficient translations between ML models, which is the core of the current paper, has not been considered in any of those works.

7 Conclusion & Perspectives

In this paper, we have studied the existence of polynomial-time translations and polynomial-space translations between ML models (and other representations) for binary classifiers. For the eleven languages considered, we have drawn *the complete map* for the relationships $\leq_p$ and $\leq_s$.

Let us now explain how one can take advantage of the results above for XAI purposes. Most of the time, when $\mathcal{C} \leq_p \mathcal{C}'$ holds, whatever the XAI query, any dedicated algorithm that is efficient enough in practice for predictors in $\mathcal{C}'$ can be turned into an algorithm for the same query that is efficient enough in practice for predictors in $\mathcal{C}$. Indeed, almost all the polynomial-time translations pointed out in the paper are in fact linear-time translations.[2] The existence of polynomial-time translations between languages for binary classifiers can also be exploited to derive intractability results for XAI queries: if $\mathcal{C} \leq_p \mathcal{C}'$ holds, then every XAI query that is NP-hard (in the broad sense) for $\mathcal{C}'$ also is NP-hard for $\mathcal{C}$. For instance, since every of the nine XAI queries considered in [Audemard *et al.*, 2021] have been shown NP-hard for RF, we can conclude that they are also NP-hard for BT, BRT, BNN, $\widehat{\text{TH}}$, BMP, TH (the classes BRT, $\widehat{\text{TH}}$, and TH were not taken into account in [Audemard *et al.*, 2021]). In such a case, when the computational complexities of the XAI queries that are considered are "mild enough", the scalability of the corresponding XAI algorithms may be sufficient in practice. Finally, the "negative results" of the form $\mathcal{C} \not\leq_s \mathcal{C}'$ that we identified are also useful. Indeed, whenever $\mathcal{C} \not\leq_s \mathcal{C}'$, using algorithms suited to XAI queries for $\mathcal{C}'$ in order to address the same XAI queries for $\mathcal{C}$ is an approach that offers no guarantee of success. Looking for XAI algorithms that are specific to $\mathcal{C}$ appears as a better option.

A perspective for future work would be to draw a version of the map for ML models that are not fully expressive. Indeed, limiting the expressiveness of the model is a way to prevent the learned predictor from overfitting the training data. In particular, for random forests, it is common to bound the depth of the decision trees used, and for boosted trees, to consider only decision stumps. We plan on studying RF and BT classifiers based on trees with bounded depth. A first direction is to compare RF classifiers to BT classifiers when both models use trees with depth bounded by a common bound. Another direction is to compare classifiers relevant to the same model but that use trees with different depths.

---

[2]Translations that are more than linear-time (for instance that between BMP and BNN) must be considered more carefully to determine whether they are efficient enough in practice.

Acknowledgments

References

[Adadi and Berrada, 2018] A. Adadi and M. Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

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

[Arrieta *et al.*, 2020] A. Barredo Arrieta, N. Díaz R., J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 58:82–115, 2020.

[Audemard *et al.*, 2020] Gilles Audemard, Frédéric Koriche, and Pierre Marquis. On Tractable XAI Queries based on Compiled Representations. In *Proc. of KR'20*, pages 838–849, 2020.

[Audemard *et al.*, 2021] Gilles Audemard, Steve Bellart, Louenas Bounia, Frédéric Koriche, Jean-Marie Lagniez, and Pierre Marquis. On the Computational Intelligibility of Boolean Classifiers. In *Proc. of KR'21*, pages 74–86, 2021.

[Audemard *et al.*, 2022] Gilles Audemard, Steve Bellart, Louenas Bounia, Frédéric Koriche, Jean-Marie Lagniez, and Pierre Marquis. Trading Complexity for Sparsity in Random Forest Explanations. In *Proc. of AAAI'22*, pages 5461–5469. AAAI Press, 2022.

[Barceló *et al.*, 2020] P. Barceló, M. Monet, J. Pérez, and B. Subercaseaux. Model Interpretability through the lens of Computational Complexity. In *Proc. of NeurIPS'20*, 2020.

[Bova *et al.*, 2014] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. Expander CNFs have Exponential DNNF Size. *CoRR*, abs/1411.1995, 2014.

[Bova *et al.*, 2016] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. Knowledge Compilation Meets Communication Complexity. In *Proc. of IJCAI'16*, pages 1008–1014, 2016.

[Breiman *et al.*, 1984] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[Breiman, 2001] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[Bryant, 1986] Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–692, 1986.

[Caruana *et al.*, 2020] R. Caruana, S. M. Lundberg, M. Túlio Ribeiro, H. Nori, and S. Jenkins. Intelligible and Explainable Machine Learning: Best Practices and Practical Challenges. In *Proc. of KDD'20*, pages 3511–3512. ACM, 2020.

[Chen and Guestrin, 2016] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proc. of KDD'16*, page 785–794, 2016.

[Darwiche and Hirth, 2020] A. Darwiche and A. Hirth. On the Reasons Behind Decisions. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI'20)*, pages 712–720, 2020.

[Darwiche and Ji, 2022] Adnan Darwiche and Chunxi Ji. On the Computation of Necessary and Sufficient Explanations. In *Proc. of AAAI'22*, pages 5582–5591, 2022.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A Knowledge Compilation Map. *J. Artif. Intell. Res.*, 17:229–264, 2002.

[Darwiche and Marquis, 2021] Adnan Darwiche and Pierre Marquis. On Quantifying Literals in Boolean Logic and its Applications to Explainable AI. *J. Artif. Intell. Res.*, 72:285–328, 2021.

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

[Darwiche, 2011] A. Darwiche. SDD: A New Canonical Representation of Propositional Knowledge Bases. In *Proc. of IJCAI'11*, pages 819–826, 2011.

[de Colnet and Marquis, 2022] Alexis de Colnet and Pierre Marquis. On the Complexity of Enumerating Prime Implicants from Decision-DNNF Circuits. In *Proc. of IJCAI'22*, pages 2583–2590, 2022.

[Du *et al.*, 2020] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Commun. ACM*, 63(1):68–77, 2020.

[Freund and Schapire, 1997] Y. Freund and R.E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[Goldmann *et al.*, 1992] Mikael Goldmann, Johan Håstad, and Alexander A. Razborov. Majority Gates VS. General Weighted Threshold Gates. *Comput. Complex.*, 2:277–300, 1992.

[Goldmann, 1997] Mikael Goldmann. On the Power of a Threshold Gate at the Top. *Inf. Process. Lett.*, 63(6):287–293, 1997.

[Goodman and Flaxman, 2017] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, Oct. 2017.

[Guidotti *et al.*, 2019] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys*, 51(5):93:1–93:42, 2019.

[Guidotti, 2022] R. Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, 2022.

[Gunning, 2019] D. Gunning. DARPA's explainable artificial intelligence (XAI) program. In *Proc. of IUI'19*, 2019.

[Håstad *et al.*, 1995] Johan Håstad, Stasys Jukna, and Pavel Pudlák. Top-Down Lower Bounds for Depth-Three Circuits. *Comput. Complex.*, 5(2):99–112, 1995.

[Huang and Darwiche, 2005] Jinbo Huang and Adnan Darwiche. DPLL with a Trace: From SAT to Knowledge Compilation. In *Proc. of IJCAI'05*, pages 156–162, 2005.

[Huang *et al.*, 2022] Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, Martin C. Cooper, Nicholas Asher, and João Marques-Silva. Tractable Explanations for d-DNNF Classifiers. In *Proc. of AAAI'22*, pages 5719–5728, 2022.

[Hubara *et al.*, 2016] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks. In *Advances in Neural Information Processing Systems 29 (NeurIPS'16)*, pages 4107–4115, 2016.

[Ignatiev *et al.*, 2019] A. Ignatiev, N. Narodytska, and J. Marques-Silva. On Validating, Repairing and Refining Heuristic ML Explanations. *CoRR*, abs/1907.02509, 2019.

[Ignatiev, 2020] A. Ignatiev. Towards Trustable Explainable AI. In *Proc. of IJCAI'20*, pages 5154–5158, 2020.

[Jukna, 2014] Stasys Jukna. Boolean Function Complexity Advances and Frontiers. *Bull. EATCS*, 113, 2014.

[Ke *et al.*, 2017] G. Ke, Q. Meng, Th. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Proc. of NeurIPS'17*, pages 3146–3154, 2017.

[Lundberg and Lee, 2017] S. Lundberg and S-I. Lee. A Unified Approach to Interpreting Model Predictions. In *Proc. of NIPS'17*, pages 4765–4774, 2017.

[Lundberg *et al.*, 2020] S. M. Lundberg, G. G. Erion, H. Chen, A. J. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S. Lee. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.*, 2(1):56–67, 2020.

[Marques-Silva and Ignatiev, 2022] J. Marques-Silva and A. Ignatiev. Delivering Trustworthy AI through Formal XAI. In *Proc. of AAAI'22*, pages 12342–12350, 2022.

[Marques-Silva, 2022] João Marques-Silva. Logic-Based Explainability in Machine Learning. *CoRR*, abs/2211.00541, 2022.

[Miller, 2019] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[Molnar, 2019] Christoph Molnar. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*. Leanpub, 2019.

[Narodytska *et al.*, 2018] Nina Narodytska, Shiva Prasad Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh. Verifying Properties of Binarized Deep Neural Networks. In *Proc. of AAAI'18*, pages 6615–6624, 2018.

[O'Donnell, 2014] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[Parmentier and Vidal, 2021] A. Parmentier and T. Vidal. Optimal Counterfactual Explanations in Tree Ensembles. In *Proc. of ICML'21*, volume 139 of *Proceedings of Machine Learning Research*, pages 8422–8431, 2021.

[Prokhorenkova *et al.*, 2018] L. O. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: unbiased boosting with categorical features. In *Proc. of NeurIPS'18*, pages 6639–6649, 2018.

[Quinlan, 1986] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986.

[Ribeiro *et al.*, 2016] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proc. of KDD'16*, pages 1135–1144, 2016.

[Ribeiro *et al.*, 2018] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-Precision Model-Agnostic Explanations. In *Proc. of AAAI'18*, pages 1527–1535, 2018.

[Rudin *et al.*, 2021] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong. Interpretable Machine Learning: Fundamental Principles and 10 Grand Challenges. *CoRR*, abs/2103.11251, 2021.

[Samek *et al.*, 2019] W. Samek, G. Montavon, A. Vedaldi, L.K. Hansen, and K.R. Müller. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.

[Schapire and Freund, 2014] R.E. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2014.

[Shi *et al.*, 2020] W. Shi, A. Shih, A. Darwiche, and A. Choi. On Tractable Representations of Binary Neural Networks. In *Proc. of KR'20*, pages 882–892, 2020.

[Shih *et al.*, 2018] Andy Shih, Arthur Choi, and Adnan Darwiche. Formal verification of Bayesian network classifiers. In *Proc. of PGM'18*, pages 427–438, 2018.

[Shih *et al.*, 2019] Andy Shih, Arthur Choi, and Adnan Darwiche. Compiling Bayesian Networks into Decision Graphs. In *Proc. of AAAI'19*, pages 7966–7974, 2019.

[Srinivasan and Chander, 2020] R. Srinivasan and A. Chander. Explanation Perspectives from the Cognitive Sciences - A Survey. In *Proc. of IJCAI'20*, pages 4812–4818, 2020.