ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230307

Rectifying Binary Classifiers

Sylvie Coste-Marquis[a] and Pierre Marquis[a,b]

[a]Univ. Artois, CNRS, CRIL, France
[b]Institut Universitaire de France, France
ORCiD ID: Sylvie Coste-Marquis https://orcid.org/0000-0003-4742-4858,
Pierre Marquis https://orcid.org/0000-0002-7979-6608

Abstract. We elaborate on the notion of *rectification of a classifier* Σ based on Boolean features, introduced in [10]. The purpose is to determine how to modify Σ when the way it classifies a given instance is considered incorrect since it conflicts with some expert knowledge T. Given Σ and T, postulates characterizing the way Σ must be changed into a new classifier $\Sigma \star T$ that complies with T were presented. We focus here on the specific case of *binary* classifiers, i.e., there is a single target concept, and any instance is classified either as positive (an element of the concept), or as negative (an element of the complementary concept). In this specific case, our main contribution is twofold: (1) we show that there is a *unique* rectification operator $\star$ satisfying the postulates, and (2) when Σ and T are Boolean circuits, we show how a classification circuit equivalent to $\Sigma \star T$ can be computed in *time linear* in the size of Σ and T; when Σ is a decision tree (resp. a random forest, a boosted tree) and T is a decision tree, a decision tree (resp. a random forest, a boosted tree) equivalent to $\Sigma \star T$ can be computed in *time polynomial* in the size of Σ and T.

1 Introduction

Though explaining is an issue considered in AI for decades, the new field of "eXplainable AI (XAI)" has emerged a couple of years ago [19]. The main objective of XAI is to make ML models less opaque. More precisely, the DARPA, at the origin of this buzz word, pointed out the following purpose of XAI: "to provide users with explanations that enable them to understand the system's overall strengths and weaknesses, convey an understanding of how it will behave in future or different situations, and *perhaps permit users to correct the system's mistakes*". Reaching this objective has gone typically through the definition of a number of explanation and/or verification tasks for various ML models, and the development and the evaluation of algorithms for addressing them. Among others, [4] has pointed out *XAI queries* that capture in formal terms, explanation and/or verification tasks. Answers to such XAI queries can be used by the user of the ML model to determine whether the model is trustable enough (or not), by confronting those answers to the own expectations of the user about the model.

Many work has been devoted for the past few years in this direction (see e.g., [1, 28, 33, 18, 36, 29, 27, 32]). The issue of *correcting the system's mistakes* has received less attention, probably because it is more tricky. Indeed, whenever the prediction made by the ML model is viewed as incorrect or, more generally, when it conflicts with the expectations of the user, a challenging issue is to figure out *how the ML model should be modified* to ensure that the prediction made will be correct afterwards, and that the predictor will comply with the expectations of the user. This calls for *XAI transformations*, i.e., methods that do not reduce to extracting useful information from the ML model (this is the purpose of XAI queries), but are designed to change the model itself.

To make things concrete and illustrate the concept of XAI transformation, let us consider the following credit scoring scenario. Alice, a bank employee, receives Bob, a customer who wants to obtain a loan. The bank management provides Alice with an AI algorithm (a predictor) to help her decide which issue to give to any loan application. Alice uses this algorithm and it recommends against granting Bob the requested loan. Alice is very surprised by the result provided by the algorithm, since she is experienced and has already granted loans to clients of the bank with precisely the same profile as Bob's, i.e., a client with low incomes but who has reimbursed a previous loan and has no debts. Alice's expertise led her not to follow the recommendation of the AI algorithm and to grant Bob the loan requested. However, Alice would like to do more to avoid that the problem encountered arises again with another client having an identical profile. She wonders what could be done to this end.

The research question tackled in the recent work [10] is relevant to Alice's concern. In this work, the authors introduced a specific XAI transformation, called *rectification*, that is suited to multi-label Boolean classifiers. Given a set $X = \{x_1, \cdots, x_n\}$ (its elements are Boolean features) and a set $Y = \{y_1, \cdots, y_m\}$, that is disjoint with X (its elements are labels, denoting classes / concepts), $\boldsymbol{X}$ is the set $\{0,1\}^n$ of all vectors over $\{0,1\}$ of size n, and $\boldsymbol{Y}$ is the set $\{0,1\}^m$ of all vectors over $\{0,1\}$ of size m. Then, a multi-label Boolean classifier simply is a mapping f from $\boldsymbol{X}$ to $\boldsymbol{Y}$, associating with each input instance (a vector $\boldsymbol{x} \in \boldsymbol{X}$ of n Boolean values assigned to the elements of X) a vector $\boldsymbol{y} \in \boldsymbol{Y}$ of m Boolean values assigned to the elements of Y. Whenever an instance $\boldsymbol{x} = (x_1, \ldots, x_n)$ is associated by the classifier with $\boldsymbol{y} = (y_1, \ldots, y_m)$ such that y_i ($i \in [m]$) is equal to 1 (resp. 0), one considers that $\boldsymbol{x}$ belongs to the class y_i (resp. does not belong to this class).

For instance, considering the previous credit scoring scenario, one may assume the following sets of Boolean features $X = \{x_1, x_2, x_3\}$ and labels $Y = \{y\}$, associated with the following semantics:

- $x_1 = 1$: has low income
- $x_2 = 1$: has reimbursed a previous loan
- $x_3 = 1$: has debts

* Corresponding Author. Email: marquis@cril.univ-artois.fr

- $y = 1$: to grant the loan

Bob is viewed as the instance $\boldsymbol{x} = (x_1 = 1, x_2 = 1, x_3 = 0)$, and if f denotes the predictor used by Alice, we have $f(\boldsymbol{x}) = 0$, meaning that the predictor suggests not to grant the loan.

A multi-label Boolean classifier f can be represented by a Boolean circuit Σ over a set of variables PS such that $X \cup Y \subseteq PS$. In such a circuit Σ, called a *classification circuit*, features and labels are both represented by propositional variables despite the fact that they correspond to distinct notions. Some pieces of knowledge supposed to be more reliable than the classification circuit Σ are also considered. They are represented as well by a Boolean circuit T over PS. The purpose of rectifying the classification circuit Σ by T is to modify Σ so that (1) the constraints imposed by T on the facts about Y that must hold under each $\boldsymbol{x}$ are respected, and (2) the resulting circuit noted $\Sigma \star T$ is still a classification circuit. A minimal change condition is taken into account; it states that the way $\boldsymbol{x}$ was classified by Σ must not be modified by the rectification process when the constraints imposed by T on the facts that hold under $\boldsymbol{x}$ are already satisfied. In the general case when Y is unconstrained, several classification circuits $\Sigma \star T$ can be found that satisfy (1) and the minimal change condition. Stated otherwise, several rectification operators $\star$ can be defined.

Whatever the operator $\star$ chosen, the value of rectification comes from the fact that each error found, i.e., each instance $\boldsymbol{x}$ badly classified by Σ, is guaranteed to be corrected if a rectification process where T indicates how $\boldsymbol{x}$ should be classified is run. Thus, if Σ does not classify $\boldsymbol{x}$ in the right way while T classifies $\boldsymbol{x}$ as it must be classified, then it is ensured $\boldsymbol{x}$ will be classified correctly by $\Sigma \star T$. This requirement is typically not met when learning-based techniques are used to (tentatively) correct badly classified instances. In such approaches (see e.g., [40]), when a new instance $\boldsymbol{x}$ is found wrongly classified by the predictor Σ, the training set used to generate Σ is updated by adding to it $\boldsymbol{x}'$, that coincides with the misclassified instance $\boldsymbol{x}$ is but associated with the right label, then a new classifier Σ' is learned from the updated dataset. However, for many ML models (including "simple ones", like decision trees or perceptrons), it can easily be the case that $\boldsymbol{x}'$ will not be classified correctly by Σ'. Stated otherwise, retraining a classifier is in general not enough to ensure that it will be corrected as one expects.

In [10], it was shown that a rectification operation amounts to a specific form of *belief change* [2]. A logical characterization of classification circuits has been pointed out and a number of postulates that rectification operators should satisfy have been presented. The authors also exhibited some operators from the rectification family, and studied the standard belief change postulates in order to determine those that are satisfied by every rectification operator satisfies, and those that are not. Especially, they proved that the families of rectification operators and those of "standard" belief change operators, namely revision operators / update operators [23, 22], are disjoint.

In this paper, we focus on the specific case of *binary classifiers*, i.e., there are only two classes, the target concept and the complementary one, so that every instance is either positive (i.e., it belongs to the target concept) or negative (i.e., it does not belong to it). This is ensured by considering that $Y = \{y\}$ is a singleton (as it it the case for the credit scoring scenario). Under this assumption, we present two new contributions. On the one hand, provided that the Boolean classifier is represented by a circuit Σ involving only variables X for the representation of instances and the concept variable y, we show that there exists a *unique rectification operator*, noted $\star$, thus providing a full characterization of rectification operators in this restricted case. Since it is unique, $\star$ coincides with the operators presented in [10] when Y is a singleton. On the other hand, we show how a classification circuit equivalent to $\Sigma \star T$ can be computed in *time linear* in the size of Σ and T, where the change formula T is given as a Boolean circuit. This result fully contrasts with the representations of rectified classifiers presented in [10], which are of size exponential in the size of X. In addition, the specific cases when Σ and T are SDD circuits [12], OBDD circuits [9], and (possibly affine) decision trees [38, 25] are analyzed. We finally show how random forests and boosted trees can be rectified in polynomial time.

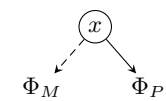
The rest of the paper is organized as follows. Some formal preliminaries are provided in Section 2. Our characterization theorem is presented in Section 3. Our representation result is given in Section 4. Other related work is discussed in Section 5. Section 6 concludes the paper.

2 Preliminaries

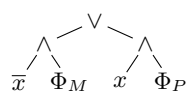
Before presenting the key definitions of the rectification setting pointed out in [10], we first need to recall a couple of notions about propositional representations.

A *Boolean circuit* Φ over a set PS of propositional variables is a DAG where internal nodes are labelled by usual connectives, $\neg, \vee, \wedge$, but may also correspond to decision nodes over variables from PS, and leaves are labelled by variables from PS or by Boolean constants $\top$ – *verum* – and $\bot$ – *falsum* (also denoted 1 and 0, respectively). The size $|\Phi|$ of Φ is the number of arcs in it. $Var(\Phi)$ the subset of variables of PS occurring in Φ. $\mathcal{L}$ is the set of all Boolean circuits over PS. When $X \subseteq PS$, $\mathcal{L}_X$ denotes the subset of $\mathcal{L}$ consisting of Boolean circuits over X.

For any node N of Φ, let Φ_N be the subcircuit of Φ rooted at N, i.e., the subgraph of Φ that consists of all the nodes and arcs that can be reached from N. When N is a decision node labelled by variable $x \in X$ in a Boolean circuit Φ, the subcircuit Φ_N of Φ given by



is viewed as a short for the Boolean circuit



A *formula* over PS simply is a Boolean circuit over PS where the underlying DAG is a tree. A *literal* is a propositional variable of PS, possibly negated, or a Boolean constant. Any propositional variable x is called a *positive literal*, and the negation of x, denoted $\neg x$ or $\overline{x}$, is called a *negative literal*. For any subset $X = \{x_1, \ldots, x_n\}$ of PS, L_X denotes the set of literals based on the variables of X. A *term* is a conjunction of literals, and a *clause* is a disjunction of literals. A *canonical term* over X is a term into which every variable of X occurs once (as such, or negated). With $\boldsymbol{X}$ the set $\{0,1\}^n$ of all vectors over $\{0,1\}$ of size n, every instance $\boldsymbol{x} \in \boldsymbol{X}$ can also be viewed as a canonical term over X, still noted $\boldsymbol{x}$, such that for every $i \in [n]$, x_i is a literal of this term if the i^{th} coordinate of $\boldsymbol{x}$ is 1 and $\overline{x_i}$ is a literal of this term otherwise.

Given a set of variables $V \subseteq PS$, an *interpretation over* V is a mapping ω from V to $\mathbb{B} = \{0,1\}$. Every interpretation over V corresponds to a unique canonical term over V, and vice versa. When

a total ordering $<$ over PS is provided, interpretations can be represented by bit vectors or by words. For instance, if $V = \{v_1, v_2\}$ such that $v_1 < v_2$, then the mapping ω such that $\omega(v_1) = 0$ and $\omega(v_2) = 1$ can be represented by the vector $(0,1) \in \{0,1\}^2$ or equivalently by the word 01. Boolean circuits are interpreted in a classical way. For a Boolean circuit $\Phi \in \mathcal{L}_V$ and an interpretation over any superset of $V = Var(\Phi)$, we use $\omega \models \Phi$ to denote the fact that ω if a *model* of Φ according to the semantics of propositional logic. That is, assigning the variables of Φ to truth values as specified by ω makes Φ true. By $[\Phi]$ we denote the *set* of models of Φ over $Var(\Phi)$. Φ is *inconsistent* if $[\Phi] = \emptyset$, and Φ is *consistent* otherwise. A Boolean circuit $\Psi \in \mathcal{L}_V$ is a *logical consequence* of a Boolean circuit $\Phi \in \mathcal{L}_V$ (denoted $\Phi \models \Psi$) if $\Phi \wedge \neg\Psi$ is inconsistent. Φ and Ψ are *logically equivalent* (denoted $\Phi \equiv \Psi$) if they have the same models over $Var(\Phi) \cup Var(\Psi)$.

Given a Boolean circuit $\Phi \in \mathcal{L}$ and a consistent term γ over PS, the *conditioning* of Φ by γ is the Boolean circuit from $\mathcal{L}$, noted $\Phi(\gamma)$, obtained by replacing in Φ every occurrence of a variable $v \in Var(\gamma)$ by $\top$ if v is a positive literal of γ and by $\bot$ if $\overline{v}$ is a negative literal of γ. Thus, conditioning Φ by γ amounts to conditioning Φ by every literal ℓ of γ in an iterative way (the ordering does not matter). Conditioning Φ_N by γ where N is a decision node labelled by x reduces to conditioning its two children by γ when $x \notin \gamma$ and $\overline{x} \notin \gamma$, and to replacing N by the conditioning of its right child (resp. left child) by γ when $x \in \gamma$ (resp. $\overline{x} \in \gamma$).

When V is a subset of PS, a Boolean circuit $\Phi \in \mathcal{L}$ is said to be *independent* of V if there is a Boolean circuit $\Psi \in \mathcal{L}$ logically equivalent to Φ such that $Var(\Psi) \cap V = \emptyset$. The *forgetting* of V in Φ, denoted $\exists V.\Phi$, is (up to logical equivalence) the *strongest logical consequence* of Φ that is independent of V (see e.g., [26]). The *projection* of Σ onto V is the forgetting of $\overline{V}$ in Σ, where $\overline{V}$ denotes the set $PS \setminus V$. $\exists V.\Phi$ can be characterized as follows:

- $\exists\emptyset.\Phi \equiv \Phi$,
- $\exists\{v\}.\Phi \equiv (\Phi(\overline{v}) \vee (\Phi(v))$,
- $\exists(V' \cup \{v\}).\Phi \equiv \exists V'.(\exists\{v\}.\Phi)$.

Let $\boldsymbol{x} \in \boldsymbol{X}$. A Boolean circuit Φ over $\mathcal{L}$ is said to *classify* $\boldsymbol{x}$ as $\boldsymbol{v}$ if and only if the Boolean circuit $\Phi(\boldsymbol{x})$ has a unique model over $V = PS \setminus X$, given by $\boldsymbol{v}$. Φ has *the XY-classification property* if and only if $Y \subseteq PS \setminus X$ and Φ classifies every $\boldsymbol{x} \in \boldsymbol{X}$. In that case, one also says that Φ is *a classification circuit*. When $PS \setminus X = \{y\}$ is a singleton, a Boolean circuit Φ is said to classify $\boldsymbol{x}$ as a *positive instance* if $\Phi(\boldsymbol{x}) \equiv y$, as a *negative instance* if $\Phi(\boldsymbol{x}) \equiv \overline{y}$, and Φ does not classify $\boldsymbol{x}$ in the remaining case.

The last notion to be made precise before defining rectification operators is the notion of *fact compliance*. A Boolean circuit $\Sigma \in \mathcal{L}$ is *fact-compliant* with a Boolean circuit $T \in \mathcal{L}$ on an instance $\boldsymbol{x}$ if and only if $\Sigma(\boldsymbol{x}) \models F(T, \boldsymbol{x})$ where

$F(T, \boldsymbol{x}) = \top$ if $T(\boldsymbol{x})$ is inconsistent,
$= \bigwedge_{\ell \in L_Y \text{ s.t. } T(\boldsymbol{x}) \models \ell} \ell$ otherwise.

When $T(\boldsymbol{x})$ is consistent, $F(T, \boldsymbol{x})$ is the conjunction of all the facts (literals) about Y that hold in $T(\boldsymbol{x})$. Accordingly, for every T and every $\boldsymbol{x}$, we have $T(\boldsymbol{x}) \models F(T, \boldsymbol{x})$.

Example 1. *Let $X = \{x_1, x_2\}$ and $Y = \{y_1, y_2\}$. Let Σ be the Boolean circuit over $X \cup Y$ given by Figure 1. Σ is logically equivalent to $(x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$. The instance $(1,1)$ corresponds to the canonical term $x_1 \wedge x_2$. Similarly, the instance $(1,0)$ corresponds to the canonical term $x_1 \wedge \overline{x_2}$. One can easily verify that $\Sigma((1,1)) \equiv y_1 \wedge y_2$, $\Sigma((1,0)) \equiv y_1 \wedge \overline{y_2}$, $\Sigma((0,1)) \equiv \overline{y_1} \wedge y_2$, and $\Sigma((0,0)) \equiv \overline{y_1} \wedge \overline{y_2}$. Thus, Σ classifies every instance $\boldsymbol{x}$, and as such, Σ is a classification circuit.*

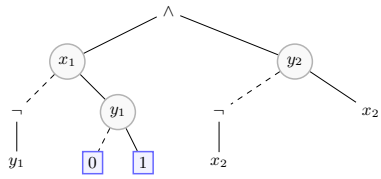


Figure 1: A classification circuit.

Now, let $T = ((x_1 \wedge x_2) \Rightarrow (y_1 \wedge y_2)) \wedge ((x_1 \wedge \overline{x_2}) \Rightarrow (y_1 \vee y_2)) \wedge ((\overline{x_1} \wedge x_2) \Rightarrow \overline{y_2}) \wedge (x_1 \vee x_2)$. T is a formula and it does not have the XY-classification property. Indeed, though T classifies $(1,1)$ (as $(1,1)$), it does not classify any of the other instances: $T((1,0))$ has three models over Y, $T((0,1))$ has two models over Y, and $T((0,0))$ is inconsistent. Finally, we have $F(T, x_1 \wedge x_2) \equiv y_1 \wedge y_2$, $F(T, x_1 \wedge \overline{x_2}) \equiv \top$, $F(T, \overline{x_1} \wedge x_2) \equiv \overline{y_2}$, and $F(T, \overline{x_1} \wedge \overline{x_2}) \equiv \top$. Thus, Σ is fact-compliant with T on every instance, but $(0,1)$.

With these definitions in hand, the notion of rectification operator can be defined as follows:

Definition 1 (rectification operator). *A* rectification operator $\star$ *is a mapping associating with two given circuits T and Σ from $\mathcal{L}$, where Σ has the XY-classification property, a circuit from $\mathcal{L}$, noted $\Sigma \star T$ and called a* rectified circuit*, such that:*

(RE1) *$\Sigma \star T$ has the XY-classification property;*
(RE2) *If Σ is fact-compliant with T on $\boldsymbol{x} \in \boldsymbol{X}$, then $(\Sigma \star T)(\boldsymbol{x}) \equiv \Sigma(\boldsymbol{x})$;*
(RE3) *For any $\boldsymbol{x} \in \boldsymbol{X}$, $(\Sigma \star T)(\boldsymbol{x}) \models F(T, \boldsymbol{x})$;*
(RE4) *If T is inconsistent, then $\Sigma \star T \equiv \Sigma$;*
(RE5) *If $\Sigma \equiv \Sigma'$ and $T \equiv T'$, then $\Sigma \star T \equiv \Sigma' \star T'$;*
(RE6) *$\Sigma \star T \equiv (\exists \overline{X \cup Y}.\Sigma) \star (\exists \overline{X \cup Y}.T)$.*

The rationale for those postulates is presented in [10]. Roughly, **(RE1)** is a closure condition: it asks that any rectified classification circuit is still a classification circuit. **(RE2)** is a minimal change condition, stating that the classification of any $\boldsymbol{x}$ as achieved by Σ should not be modified by the rectification operation whenever Σ is fact-compliant with T on $\boldsymbol{x}$. **(RE3)** is a success condition: it demands that the rectified circuit $\Sigma \star T$ is fact-compliant with T on every $\boldsymbol{x}$. **(RE4)** is a non-triviality condition; it deals with the case when T is inconsistent; in such a situation, a minimal change of Σ consists in not modifying it at all. **(RE5)** is a standard principle of irrelevance of syntax. Finally, **(RE6)** is a relevance condition: it states that the result of rectifying Σ by T must not depend on the variables outside $X \cup Y$.

Example 2. *Let us consider again the classification circuit Σ and the formula T presented at Example 1. Since Σ is fact-compliant with T on every instance, but $(0,1)$, **(RE2)** imposes that $(\Sigma \star T)((1,1)) \equiv y_1 \wedge y_2$, $(\Sigma \star T)((1,0)) \equiv y_1 \wedge \overline{y_2}$, and $(\Sigma \star T)((0,0)) \equiv \overline{y_1} \wedge \overline{y_2}$. **(RE3)** requires that $(\Sigma \star T)((0,1)) \models \overline{y_2}$. Finally, **(RE1)** ensures that $(\Sigma \star T)((0,1)) \equiv \overline{y_1} \wedge \overline{y_2}$ or $(\Sigma \star T)((0,1)) \equiv y_1 \wedge \overline{y_2}$. Accordingly, the classification $\overline{y_1} \wedge y_2$ of the instance $(0,1)$ as achieved by Σ can be rectified in* two *distinct ways in order to enforce that $\overline{y_2}$ holds. Indeed, since no independence assumptions are made about the labels of Y, it can make sense to change the truth value of label y_1 when changing the truth value of label y_2. The situation is similar to what happens in belief revision, where revising $\overline{y_1} \wedge y_2$ by $\overline{y_2}$ may lead to $y_1 \wedge \overline{y_2}$ without questioning the satisfaction of the revision postulates.*

3 A Characterization Theorem

Unlike what happens in the general case (as exemplified above), there is a *unique operator* $\star$ satisfying the rectification postulates in the restricted case when Y contains a single label:

Proposition 1. *If $\mathcal{L}$ is a language of Boolean circuits over a set of propositional variables $PS = X \cup \{y\}$, then there is a unique rectification operator $\star$.*

Proof. First of all, because of postulate **(RE5)**, we know that the syntactic representations of Σ and T does not play any role in the definition of $\Sigma \star T$ ($\star$ is syntax-independent). Now, when $Y = \{y\}$ is a singleton, we can mainly get rid of y in the representation of the classifier Σ and consider it as implicit (this is usually done in binary classifiers for the sake of simplicity). Indeed, by definition (see Definition 4 in [10]), Σ is a classification circuit if and only if there exists a circuit Σ_X from $\mathcal{L}_X$ such that $\Sigma \equiv \Sigma_X \Leftrightarrow y$. The models of Σ_X are precisely those truth assignments $\boldsymbol{x}$ over X such that $\Sigma(\boldsymbol{x}) \equiv y$. Because of postulate **(RE1)**, defining $\Sigma \star T$ just amounts to pointing out a circuit Σ_X^T from $\mathcal{L}_X$, so that $\Sigma \star T \equiv \Sigma_X^T \Leftrightarrow y$. We now show that, given Σ and T, the rectification postulates ensure the existence of a unique circuit Σ_X^T up to logical equivalence. Let $\boldsymbol{x} \in \boldsymbol{X}$. Since T is a circuit from $\mathcal{L}$ and $Y = \{y\}$, $T(\boldsymbol{x})$ is equivalent to y, $\overline{y}$, $\top$, or $\bot$. Accordingly, $F(T, \boldsymbol{x})$ is equivalent to $T(\boldsymbol{x}) \equiv y$, $T(\boldsymbol{x}) \equiv \overline{y}$, or to $\top$, so that $F(T, \boldsymbol{x})$ is equivalent to $\top$ precisely when it is not equivalent to $T(\boldsymbol{x})$. Because of postulate **(RE1)**, $(\Sigma \star T)(\boldsymbol{x})$ is equivalent to y or to $\overline{y}$. By definition, Σ is fact-compliant with T on $\boldsymbol{x}$ precisely when $F(T, \boldsymbol{x})$ is equivalent to $\top$ or $\Sigma(\boldsymbol{x})$ is equivalent to $T(\boldsymbol{x})$, and in this case, because of **(RE2)**, one must have $(\Sigma \star T)(\boldsymbol{x}) \equiv \Sigma(\boldsymbol{x})$. Thus, for any $\boldsymbol{x} \in \boldsymbol{X}$ such that $F(T, \boldsymbol{x})$ is equivalent to $\top$ or $\Sigma(\boldsymbol{x})$ is equivalent to $T(\boldsymbol{x})$, $\boldsymbol{x}$ is a model of Σ_X^T if and only if $\boldsymbol{x}$ is a model of Σ_X. The remaining case, i.e., when $F(T, \boldsymbol{x})$ is not equivalent to $\top$ and $\Sigma(\boldsymbol{x})$ is not equivalent to $T(\boldsymbol{x})$, can be simplified as $\Sigma(\boldsymbol{x})$ is not equivalent to $F(T, \boldsymbol{x}) \equiv T(\boldsymbol{x})$. Because of postulate **(RE3)**, in this case, the class of $\boldsymbol{x}$ must be switched (from positive to negative, or vice-versa), so that $\boldsymbol{x}$ is a model of Σ_X^T if and only if $\boldsymbol{x}$ is not a model of Σ_X. This shows that Σ_X^T is unique up to logical equivalence, or equivalently that there exists a unique rectification operator $\star$. Note that $\star$ trivially satisfies **(RE4)** since if T is inconsistent, $F(T, \boldsymbol{x})$ is equivalent to $\top$ for every $\boldsymbol{x} \in \boldsymbol{X}$, and $\star$ trivially satisfies **(RE6)** since $\mathcal{L}$ is built solely upon variables from X and Y (thus, $\exists \overline{X \cup Y}.\Sigma$ is equivalent to Σ and $\exists \overline{X \cup Y}.T$ is equivalent to T). □

Example 3. *As a matter of illustration, let us consider again the loan allocation scenario with Alice and Bob, as sketched in the introduction. Let us suppose that the predictor f furnished by the bank labels an instance positive when it corresponds to a customer who has high incomes ($\overline{x_1}$) but has not reimbursed a previous loan ($\overline{x_2}$), or (which looks more risky) a customer who has low incomes (x_1) and has some debts (x_3). Suppose also that Alice's expertise consists of two decision rules stating that if a customer has low incomes but no debts, the loan can be granted, while if a customer has not reimbursed a previous loan, the loan should not be granted.*

Formally, the predictor f can be represented by the classification circuit $\Sigma = \Sigma_X \Leftrightarrow y$ where $\Sigma_X = (\overline{x_1} \wedge \overline{x_2}) \vee (x_1 \wedge x_3)$. Alice's expertise can be represented by the formula $T = T_1 \wedge T_2$ with $T_1 = (x_1 \wedge \overline{x_3}) \Rightarrow y$ and $T_2 = \overline{x_2} \Rightarrow \overline{y}$ encode Alice's decision rules. For every instance $\boldsymbol{x} \in \boldsymbol{X}$, Table 1 indicates from left to right, whether or not Σ classifies $\boldsymbol{x}$ as positive (this is the case precisely when $\Sigma(\boldsymbol{x}) \equiv y$), the constraint imposed by T on the way $\boldsymbol{x}$ should be classified (i.e., as positive when $T(\boldsymbol{x}) \equiv y$ and as negative when $T(\boldsymbol{x}) \equiv \overline{y}$), the facts $F(T, \boldsymbol{x})$ that hold in T under $\boldsymbol{x}$, and finally whether or not $\Sigma \star T$ classifies $\boldsymbol{x}$ as positive (this is the case precisely when $(\Sigma \star T)(\boldsymbol{x}) \equiv y$).

$\boldsymbol{x}$	$\Sigma(\boldsymbol{x})$	$T(\boldsymbol{x})$	$F(T, \boldsymbol{x})$	$(\Sigma \star T)(\boldsymbol{x})$
000	y	$\overline{y}$	$\overline{y}$	$\overline{y}$
001	y	$\overline{y}$	$\overline{y}$	$\overline{y}$
010	$\overline{y}$	$\top$	$\top$	$\overline{y}$
011	$\overline{y}$	$\top$	$\top$	$\overline{y}$
100	$\overline{y}$	$\bot$	$\top$	$\overline{y}$
101	y	$\overline{y}$	$\overline{y}$	$\overline{y}$
110	$\overline{y}$	y	y	y
111	y	$\top$	$\top$	y

Table 1: The credit scoring scenario, with Alice and Bob.

The instance $\boldsymbol{x} = (x_1 = 1, x_2 = 1, x_3 = 0)$ in orange in the table corresponds to Bob. The classification circuit considered at start classifies $\boldsymbol{x}$ as a negative instance ($\Sigma(\boldsymbol{x}) \equiv \overline{y}$). Contrastingly, the rectified classification circuit $\Sigma \star T$ once T has been taken into account classifies $\boldsymbol{x}$ as positive ($(\Sigma \star T)(\boldsymbol{x}) \equiv y$), as it is expected. One can observe by looking at the table that no specific assumptions are made about what T must say about y under a partial assignment $\boldsymbol{x}$. Thus, the information conveyed by T about $\boldsymbol{x}$ can be trivial, i.e., equivalent to $\top$ (this is the case for instance for $\boldsymbol{x} = (x_1 = 0, x_2 = 1, x_3 = 0)$) or contradictory – equivalent to $\bot$ – (this is the case for $\boldsymbol{x} = (x_1 = 1, x_2 = 0, x_3 = 0)$ since Alice's decision rules T_1 and T_2 are both triggered under this assignment, and those rules have conflicting conclusions).

Note that when Y is a singleton, for every $\boldsymbol{x} \in \boldsymbol{X}$ such that $T(\boldsymbol{x})$ is consistent, we have $T(\boldsymbol{x}) \equiv F(T, \boldsymbol{x})$. Then **(RE3)** shows immediately that for every $\boldsymbol{x} \in \boldsymbol{X}$ such that $T(\boldsymbol{x})$ is consistent, $\Sigma \star T$ is *knowledge-compliant* with T on $\boldsymbol{x}$, i.e., $(\Sigma \star T)(\boldsymbol{x}) \models T(\boldsymbol{x})$ [10].

4 Representing Rectified Classifiers

Some rectification operators have been put forward in [10]. Among them is the following $\star_D$ operator:

Definition 2 ($\star_D$). *Let $\circ_D$ denote Dalal revision operator [11].*[1] *Let $\star_D$ be the mapping associating with $T \in \mathcal{L}$ and a classification circuit $\Sigma \in \mathcal{L}$, a classification circuit $\Sigma \star_D T \in \mathcal{L}$ such that*

$$\Sigma \star_D T \equiv \bigvee_{\boldsymbol{x} \in \boldsymbol{X}} \boldsymbol{x} \wedge (\Sigma \star_D T)(\boldsymbol{x})$$

where for any $\boldsymbol{x} \in \boldsymbol{X}$, $(\Sigma \star_D T)(\boldsymbol{x}) = \Sigma(\boldsymbol{x}) \circ_D F(T, \boldsymbol{x})$.

It was already observed that $\star_D$ coincides with other rectification operators pointed out in [10] when Y is a singleton. Thanks to Proposition 1, we now know more: there is no rectification operator $\star$ that would be different of $\star_D$. Accordingly, the definition of $\star_D$ induces in a straightforward way a characterization result for the class of rectification operators when $|Y| = 1$.

However, the definition of $\star_D$ above is *not convenient at all from a representation perspective* since the representation $\bigvee_{\boldsymbol{x} \in \boldsymbol{X}} \boldsymbol{x} \wedge (\Sigma \star_D T)(\boldsymbol{x})$ of the rectified classifier $\Sigma \star_D T$ is of size exponential in $|X|$. In the following, we explain how a much more compact representation of $\Sigma \star T$ can be derived. This representation can be computed

[1] Given two propositional representations φ and α, the models of $\varphi \circ_D \alpha$ consist of the models of α which are as close as possible to φ w.r.t. Hamming distance.

in *time linear* in $|\Sigma|$ and of $|T|$, and its size also is linear in the size of $|\Sigma|$ and of $|T|$. Remember that when $Y = \{y\}$, because of **(RE1)**, one knows that there exists a circuit Σ_X^T from $\mathcal{L}_X$ so that $\Sigma \star T \equiv \Sigma_X^T \Leftrightarrow y$. Thus, generating a circuit representing $\Sigma \star T$ boils down to generating a circuit representing Σ_X^T.

To do so, one first need to make precise the instances that are classified by T as positive, and those that are classified by T as negative.

Proposition 2. *Let $\boldsymbol{x} \in \boldsymbol{X}$ and $T \in \mathcal{L}$. T classifies $\boldsymbol{x}$ as*

- *a positive instance if $\boldsymbol{x} \models T(y) \wedge \neg T(\overline{y})$;*
- *a negative instance if $\boldsymbol{x} \models T(\overline{y}) \wedge \neg T(y)$.*

Proof. Let us consider the case of positive instances (the other case is similar). By definition, T classifies $\boldsymbol{x}$ as a positive instance if and only if $T(\boldsymbol{x}) \equiv y$. This means precisely that the assignment $\omega_{\boldsymbol{x},y}$ that coincides with $\boldsymbol{x}$ over X and sets y to true is a model of T and that the assignment $\omega_{\boldsymbol{x},\overline{y}}$ that coincides with $\boldsymbol{x}$ over X and sets y to false is not a model of T (if both $\omega_{\boldsymbol{x},y}$ and $\omega_{\boldsymbol{x},\overline{y}}$ were models of T, then we would have $T(\boldsymbol{x}) \equiv \top$, and if none of $\omega_{\boldsymbol{x},y}$ and $\omega_{\boldsymbol{x},\overline{y}}$ were models of T, then we would have $T(\boldsymbol{x}) \equiv \bot$). But $\omega_{\boldsymbol{x},y} \models T$ precisely means that $\boldsymbol{x} \models \exists\{y\}.(T \wedge y)$, or equivalently that $\boldsymbol{x} \models T(y)$. And similarly, $\omega_{\boldsymbol{x},\overline{y}} \not\models T$ precisely means that $\boldsymbol{x} \not\models \exists\{y\}.(T \wedge \overline{y})$, or equivalently that $\boldsymbol{x} \not\models T(\overline{y})$. Finally, since $T(\overline{y})$ is a circuit from $\mathcal{L}_X$ and $\boldsymbol{x}$ is an assignment over X, we have $\boldsymbol{x} \not\models T(\overline{y})$ if and only if $\boldsymbol{x} \models \neg T(\overline{y})$. This concludes the proof. □

On this basis, the following representation of Σ_X^T can be derived:

Proposition 3. *Let $\Sigma_X \in \mathcal{L}_X$ and $T \in \mathcal{L}$. We have*

$$\Sigma_X^T \equiv (\Sigma_X \wedge \neg(T(\overline{y}) \wedge \neg T(y))) \vee (T(y) \wedge \neg T(\overline{y})).$$

Proof. Proposition 1 ensures that Σ_X^T is unique up to logical equivalence. Then, the result comes directly from the identification of the only two reasons according to which an instance $\boldsymbol{x} \in \boldsymbol{X}$ must be classified as positive by the rectified classifier (i.e., when $\boldsymbol{x}$ is a model of Σ_X^T):

- Because of **(RE2)**, $\boldsymbol{x}$ is a model of Σ_X^T when $\boldsymbol{x}$ is a model of Σ_X and the change formula T does not classify $\boldsymbol{x}$ as negative (hence, $\Sigma_X \Leftrightarrow y$ is fact-compliant with T on $\boldsymbol{x}$). By construction, given Proposition 2, every such model is a model of $\Sigma_X \wedge \neg(T(\overline{y}) \wedge \neg T(y))$.
- Because of **(RE3)**, $\boldsymbol{x}$ is a model of Σ_X^T when T classifies $\boldsymbol{x}$ as a positive instance. By construction, given Proposition 2, every such model is a model of $T(y) \wedge \neg T(\overline{y})$.

□

The rationale of this characterization of Σ_X^T is as follows. For an instance $\boldsymbol{x}$ to be classified as positive by the rectified classification circuit, it must be the case that either T consistently asks for it (this corresponds to the disjunct $T(y) \wedge \neg T(\overline{y})$), or that the classification circuit considered at start classifies $\boldsymbol{x}$ as positive, provided that T does not consistently ask $\boldsymbol{x}$ to be classified as negative (this corresponds to the disjunct $\Sigma_X \wedge \neg(T(\overline{y}) \wedge \neg T(y))$). Such a construction is reminiscent to the representation of STRIPS-like actions using propositional formulae, thus asking to make precise each situation where a fluent holds so as to handle the frame problem.

From Proposition 3, since the conditioning transformation on circuits can be achieved in linear time, $\Sigma_X^T \Leftrightarrow y$ (where Σ_X^T is provided by Proposition 3) is a circuit of $\mathcal{L}$ equivalent to $\Sigma \star T$ and computable in time linear in $|\Sigma| + |T|$. Its size is also linear in $|\Sigma| + |T|$, as expected.

Example 4. *Let us consider Σ and T as in Example 3. We have $T(y) \equiv x_2$ and $T(\overline{y}) \equiv \overline{x_1} \vee x_3$. Thus, we get*

$$\Sigma_X^T \equiv (\underbrace{(\overline{x_1} \wedge \overline{x_2}) \vee (x_1 \wedge x_3)}_{\Sigma_X} \wedge \neg(\underbrace{(\overline{x_1} \vee x_3)}_{T(\overline{y})} \wedge \neg \underbrace{x_2}_{T(y)}))$$
$$\vee(\underbrace{x_2}_{T(y)} \wedge \neg(\underbrace{(\overline{x_1} \vee x_3)}_{T(\overline{y})})).$$

This circuit can be simplified as $\Sigma_X^T \equiv x_1 \wedge x_2$. One can check in Table 1 (rightmost colum) that the models of Σ_X^T are precisely those $\boldsymbol{x} \in \boldsymbol{X}$ such that $(\Sigma \star T)(\boldsymbol{x}) \equiv y$. Stated otherwise, for the rectified classification circuit $\Sigma \star T$, the clients for which a loan can be granted are those having low incomes provided that they have reimbursed a previous loan.

If Σ_X and T are formulae (and not general Boolean circuits) in Proposition 3, then the resulting characterization of Σ_X^T also is a formula (indeed, conditioning a formula leads to a formula). Thus, as a direct corollary to Proposition 3, we get:

Corollary 1. *Whenever Σ_X and T belongs to a class $\mathcal{C}$ of circuits that offers in polynomial time the transformations of negation ($\neg\mathbf{C}$), bounded conjunction ($\wedge\mathbf{BC}$), and bounded disjunction ($\vee\mathbf{BC}$) [13], a representation of Σ_X^T in $\mathcal{C}$ can be* derived in polynomial time *from Σ_X and T.*

Notably, focusing on a restricted class of circuits $\mathcal{C}$ is not mandatory for ensuring tractable classification: when Σ_X^T is in $\mathcal{L}_X$, deciding whether $\boldsymbol{x} \in \boldsymbol{X}$ is classified as positive by $\Sigma \star T$ amounts to testing whether $\boldsymbol{x}$ is a model of Σ_X^T, and such a model checking test can be done in time linear in the size of the input. However, considering specific classes of circuits can be useful from an XAI perspective (see e.g., [4, 5, 3, 37, 21]).

Among the classes of circuits offering $\neg\mathbf{C}$, $\wedge\mathbf{BC}$, and $\vee\mathbf{BC}$ are SDD, the class of sentential decision diagrams [12], OBDD, the class of ordered binary decision diagrams [9], but also DT, the class of decision trees, and more generally ADT, the class of affine decision trees [25]. Any Boolean circuit can be represented in SDD, OBDD, ADT and DT. Thus, considering those languages for representing the change formula T that triggers the rectification operation allows us to accept as input any possible T (up to logical equivalence). Of course, it is not the case that every Boolean circuit T has a representation in SDD, OBDD, ADT or DT that is of size polynomial in $|T|$, but "simple" change formulae T (e.g., clauses or terms) can be turned in linear time into equivalent representations in SDD, OBDD, ADT, and DT. For instance, a decision rule like $(x_1 \wedge \overline{x_3}) \Rightarrow y$ (equivalent to the clause $\overline{x_1} \vee x_3 \vee y$) that is entailed by the formula T considered in Example 3 could be easily handled.

The case of DT is of particular interest since it corresponds to a well-known ML model [8, 31], that also serves as a key component of other ML models, especially random forests RF [7] and boosted trees [15]. Rectifying a (possibly weighted) random forest simply amounts to rectifying every decision tree in it while keeping the weight of each tree as it is, thus the rectification process is also tractable for (possibly weighted) random forests. The case of boosted trees is a bit more tricky, but in the case of binary classification, every boosted tree (whatever its form) can be turned in polynomial time into an equivalent weighted random forest. This is easy to figure out for Adaboost-style boosted trees, which are similar to weighted random forests, except that the leaves are labelled by $+1$ and -1 instead of 1 and 0. In this case, every tree Σ_X with weight w can be replaced by two decision trees Σ_X^{+1} and Σ_X^{-1} where Σ_X^{+1} and Σ_X^{-1} are copies of Σ_X

where every leaf labelled by 1 (resp. -1) in Σ_X corresponds to a leaf labelled by 1 (resp. 0) in Σ_X^{+1}, and every leaf labelled by 1 (resp. -1) in Σ_X corresponds to a leaf labelled by 0 (resp. 1) in Σ_X^{-1}, while the weight of Σ_X^{+1} is w and the weight of Σ_X^{-1} is $-w$. In the case of XGBoost-style boosted trees, the trees are regression trees, i.e., with leaves are labelled by weights (real numbers). Then every tree Σ_X with leaves labelled by weights $w_1, \ldots, w_p$ can be replaced by p weighted decision trees $\Sigma_X^{w_1}, \ldots, \Sigma_X^{w_p}$ where the weight associated with $\Sigma_X^{w_i}$ ($i \in [p]$) is w_i and $\Sigma_X^{w_i}$ is a copy of Σ_X where every leaf labelled by w_i in Σ_X corresponds to a leaf labelled by 1 in $\Sigma_X^{w_i}$, while all the remaining leaves of Σ_X correspond to leaves labelled by 0 in $\Sigma_X^{w_i}$. For more on translations between ML models, see [14].

Given the significance of tree-based models (decision trees, random forests, and boosted trees) in ML and their performance when dealing with tabular data [6, 17], the availability of polynomial-time rectification algorithms for such models is a *noteworthy consequence* of Proposition 3.

Example 5. *Considering Example 3 again, let us illustrate how a decision tree classifier equivalent to $\Sigma \star T$ can be generated in polynomial time from Σ and T.*

Starting with a decision tree Σ over $X \cup \{y\}$, a decision tree over X equivalent to Σ_X (given at Figure 2 (left)) can be obtained by conditioning Σ by y since $\Sigma_X \equiv \Sigma(y)$. Conditioning a decision tree by a literal v (resp. $\overline{v}$) amounts to replacing in the tree every decision node over variable v by its right (resp. left) child. Using the conditioning transformation, from the decision tree T over $X \cup \{y\}$ at Figure 2 (right), one can derive efficiently decision trees for $T(\overline{y})$ and $T(y)$ (in Figure 2 (right), they are the subtrees rooted at nodes $T(\overline{y})$ and $T(y)$).

On this basis, deriving decision trees equivalent to $T(\overline{y}) \wedge \neg T(y)$ and $T(y) \wedge \neg T(\overline{y})$, as reported on Figures 3 (left) and 3 (right) (respectively), requires to be able to negate and to conjoin decision trees. Negating a tree consists in replacing each of its 1-leaves by a 0-leaf, and vice-versa. Conjoining two decision trees consists in replacing every 1-leaf of the first tree by a copy of the second tree. In the general case, the conjunction operation may lead to a decision tree that is not simplified, because it is not read-once and may include decision nodes having two identical children [38]. However, such a tree can be simplified in linear time into an equivalent tree, using the following rules whenever applicable: on the one hand, every decision node over a variable x_i can be replaced by its left (resp. right) child when it is itself the left (resp. right) child of a decision node over x_i or when it has an ancestor satisfying this property; on the other hand, a decision node having two identical children can be replaced by any of its two children.

Figure 4 (left) presents a decision tree equivalent to $\Sigma_X \wedge \neg(T(\overline{y}) \wedge \neg T(y))$ and obtained by conjoining the decision tree of Σ_X given in Figure 2 (left) with the negation of the decision tree of $T(\overline{y}) \wedge \neg T(y)$ given in Figure 3 (left). Figure 4 (right) illustrates the effect of the simplification process.

Figure 5 (left) presents a decision tree equivalent to $(\Sigma_X \wedge \neg(T(\overline{y}) \wedge \neg T(y))) \vee (T(y) \wedge \neg T(\overline{y}))$, obtained by disjoining the decision tree at Figure 4 (right) with the decision tree at Figure 3 (left). This is achieved by replacing every 0-leaf of the first tree by a copy of the second tree. The resulting tree is not simplified, and Figure 5 (right) presents an equivalent, yet simplified decision tree (obtained by running the simplification procedure sketched above). By construction, it is equivalent to Σ_X^T. As expected, one recovers here the condition $x_1 \wedge x_2$ characterizing the positive instances w.r.t. the rectified classification circuit: the clients for which a loan can be granted are those having low incomes provided that they have reimbursed a previous loan. Finally, a decision tree equivalent to $\Sigma \star T$ can be generated in linear time from Σ_X^T by replacing every 1-leaf (resp. 0-leaf) by a decision node over y, with a 0-leaf (resp. 1-leaf) as left child and a 1-leaf (resp. 0-leaf) as right child.

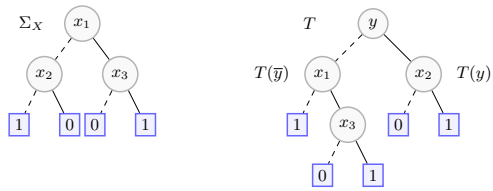


Figure 2: A decision tree representing Σ_X (left) and a decision tree representing T (right). The subtrees of T rooted at nodes $T(\overline{y})$ and $T(y)$, respectively, are decision trees representing $T(\overline{y})$ and $T(y)$.

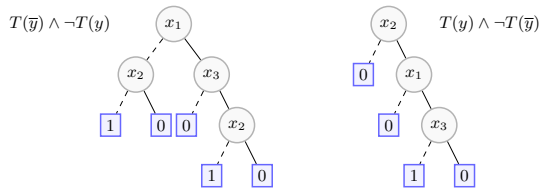


Figure 3: A decision tree representing $T(\overline{y}) \wedge \neg T(y)$ (left) and a decision tree representing $T(y) \wedge \neg T(\overline{y})$ (right).

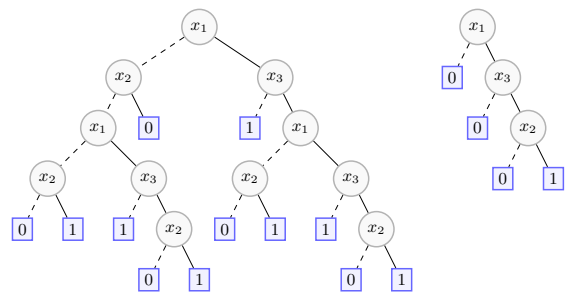


Figure 4: A decision tree representing $\Sigma_X \wedge \neg(T(\overline{y}) \wedge \neg T(y))$ (left) and this decision tree once simplified (right).

Dealing with constrained decision-functions In the previous sections, the atomic Boolean conditions represented by the propositional symbols from X are supposed to be *logically independent*. This simply means that every $\boldsymbol{x} \in \boldsymbol{X}$ is considered as feasible. However, when the classifier used is based on trees, the Boolean conditions labelling decision nodes are, in general, not independent. This is because the Boolean conditions used to describe $\boldsymbol{x} \in \boldsymbol{X}$ are usually generated from attributes A that are not of Boolean type. For example, Bob's annual income primarily is a number, let us say $A = \$15k$. A tree-based classifier Σ_X may contain a Boolean condition like $x_1 = (A < 20)$ ("has very low income"), but Σ could also contain other Boolean conditions about A, like $x_4 = (A < 30)$ ("has low income"). Clearly enough, those two conditions are not independent

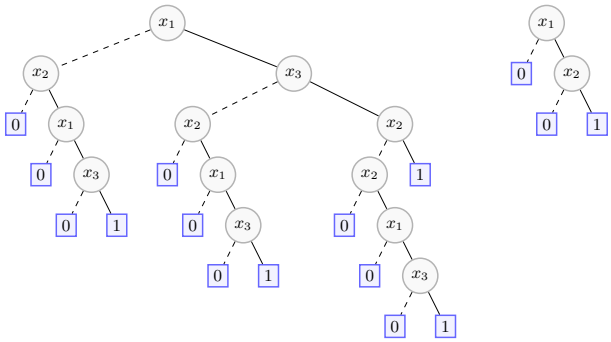


Figure 5: A decision tree representing Σ_X^T (left) and this decision tree once simplified (right).

since there is no instance $\boldsymbol{x} \in \boldsymbol{X}$ satisfying x_1 and not satisfying x_4. Taking into account the logical links existing between Boolean conditions occurring in Σ_X calls for considering a domain theory φ over X (here, $\varphi \equiv x_1 \Rightarrow x_4$), thus to consider a *constrained decision-function* (Σ_X, φ) instead of Σ_X alone [16].

While the presence of a domain theory φ usually has an impact on answering XAI queries (especially, on the local explanations – abductive or contrastive – for an instance that are expected [16, 39]), it does not change the way rectification is achieved. Indeed, the role of φ for the classification task is only to filter out impossible instances. Thus, given a classifier $\Sigma = \Sigma_X \Leftrightarrow y$ and a domain theory φ, there are three (mutually exclusive) cases: (1) if $\boldsymbol{x} \not\models \varphi$, then $\Sigma(\boldsymbol{x})$ is undefined; (2) if $\boldsymbol{x} \models \Sigma_X \wedge \varphi$, then Σ classifies $\boldsymbol{x}$ as positive; (3) if $\boldsymbol{x} \models \neg\Sigma_X \wedge \varphi$, then Σ classifies $\boldsymbol{x}$ as negative. When some expert knowledge T is considered, the corresponding rectified constrained decision-function is simply given by (Σ_X^T, φ) such that $\Sigma \star T = \Sigma_X^T \Leftrightarrow y$ and again there are three (mutually exclusive) cases: (1) if $\boldsymbol{x} \not\models \varphi$, then $(\Sigma \star T)(\boldsymbol{x})$ is undefined; (2) if $\boldsymbol{x} \models \Sigma_X^T \wedge \varphi$, then $\Sigma \star T$ classifies $\boldsymbol{x}$ as positive; (3) if $\boldsymbol{x} \models \neg\Sigma_X^T \wedge \varphi$, then $\Sigma \star T$ classifies $\boldsymbol{x}$ as negative. That mentioned, it is easy to check that for every $\boldsymbol{x} \in [\varphi]$, $(\Sigma_X \Leftrightarrow y) \wedge \varphi$ and $\Sigma_X \Leftrightarrow y$ classify $\boldsymbol{x}$ in the same way and $T \wedge \varphi$ and T classify $\boldsymbol{x}$ in the same way. As a consequence, φ can be used to *simplify* Σ_X and T in the sense that replacing Σ_X by any Σ'_X and T by any T' such that $(\Sigma_X \Leftrightarrow y) \wedge \varphi \equiv (\Sigma'_X \Leftrightarrow y) \wedge \varphi$ and $T \wedge \varphi \equiv T' \wedge \varphi$ will lead to equivalent rectified constrained decision-functions. Accordingly, for the running example, if $\varphi = x_3 \Rightarrow x_1$ was supposed (in words, "everyone who has debts has low income"), $\Sigma_X \equiv (\overline{x_1} \wedge \overline{x_2}) \vee (x_1 \wedge x_3)$ could be replaced by the simpler formula $\Sigma'_X = (\overline{x_1} \wedge \overline{x_2}) \vee x_3$.

5 Other Related Work

XAI transformations that differ from rectification have been considered so far in the literature. One of them consists in *retraining* (i.e., learning again): when a prediction is viewed as incorrect, a simple approach that can be used to fix it is to add the corresponding instance (together with the right prediction) to the training set, then learning a new model. The main advantage of such an approach is that there is no need to design a specific correction algorithm for the ML model under consideration. The main drawback is that most of the time there is no insurance that the approach is effective (even for simple ML models as decision trees), in the sense that the prediction provided by the new model for the instance at hand can still be wrong.

The retraining transformation has been considered in [30] as an approach to possibly repair multi-layer perceptrons (MLPs) when their safety is not considered as sufficient to be used. The goal was to guarantee that each prediction made (a regression value) ranges within specific bounds. An abstraction of the input MLP as a Boolean combination of linear arithmetic constraints has been used not only to estimate how safe the MLP is (i.e., to compute a superinterval of the safety interval of the MLP) but also to identify spurious counterexamples that can be exploited to train a new MLP offering tighter safety bounds). Both the ML issue (regression vs. classification), the ML model (MLP vs. trees), and the way the repair is triggered (safety interval vs. expert knowledge), make this work quite different from our own one. In the same direction, [40] presents an approach for improving the decisions made by a classifier (whatever the underlying ML model), thanks to the use of a reasoning module. The retraining transformation is leveraged to this purpose. Like [30], the approach proposed in [40] is not specific to Boolean classifiers (unlike our approach). The lack of guarantee that required change is achieved makes those approaches significantly different from rectification.

More recently, [34] has pointed out another XAI transformation suited to Boolean classifiers, namely *editing*. The purpose of editing is to determine how a Boolean classifier should be modified when new pieces of evidence must be incorporated. In a rectification process, the class associated to an instance by the classifier is preserved unless this classification conflicts with the expert knowledge, while this is not the case when classifier is edited in the light of new examples (or counter-examples). This makes editing more suited to incremental learning than rectification. Like [10], [34] follows an axiomatic perspective: the main objective is to delineate the rational ways of editing a Boolean classifier using postulates, not to provide algorithms to implement specific editing operations. This is another significant difference w.r.t. our own work.

Finally, it is worth noting that the retraining transformation and the editing transformation are more liberal than rectification: applying those transformations may lead to change the classification of instances not involved in the change operation. This never happens when rectification is used because of **(RE2)**.

6 Conclusion

In this paper, we have presented a characterization theorem for the unique rectification operator $\star$ obtained when dealing with mono-label Boolean classifiers. We have explained how a classification circuit equivalent to $\Sigma \star T$ can be computed in time linear in the size of Σ and T. We have also shown that a decision tree equivalent to $\Sigma \star T$ can be computed in time polynomial in the size of Σ and T when Σ is a decision tree or, more generally, a tree-based classifier (a random forest, a boosted tree) and T can be represented as a decision tree.

In our work, one started with the basic assumption that the available background knowledge T is more reliable than the classification circuit Σ. We believe that it is a reasonable assumption for many scenarios. Especially, the assumption is similar to the one considered in AGM belief revision (primacy of the new information). That mentioned, just like AGM belief revision is not suited to every revision issue (semi-revision [20], promotion [35], or improvement [24], can be used when the assumption does not hold), it would be interesting to determine how to relax the basic assumption and deal with pieces of expert knowledge that might be faulty or conflicting. Another perspective for further research is to compare the accuracy increase achieved by correcting tree-based classifiers using retraining with those obtained via rectification.

Acknowledgements

The authors would like to thank 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

[1] A. Adadi and M. Berrada, 'Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)', *IEEE Access*, **6**, 52138–52160, (2018).
[2] C. E. Alchourrón, P. Gärdenfors, and D. Makinson, 'On the logic of theory change: Partial meet contraction and revision functions', *Journal of Symbolic Logic*, **50**, 510–530, (1985).
[3] M. Arenas, P. Barceló, L. E. Bertossi, and M. Monet, 'The tractability of SHAP-score-based explanations for classification over deterministic and decomposable boolean circuits', in *Proc. of AAAI'21*, pp. 6670–6678, (2021).
[4] G. Audemard, F. Koriche, and P. Marquis, 'On tractable XAI queries based on compiled representations', in *Proc. of KR'20*, pp. 838–849, (2020).
[5] P. Barceló, M. Monet, J. Pérez, and B. Subercaseaux, 'Model interpretability through the lens of computational complexity', in *Proc. of NeurIPS'20*, (2020).
[6] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, 'Deep neural networks and tabular data: A survey', *CoRR*, **abs/2110.01889**, (2021).
[7] L. Breiman, 'Random forests', *Machine Learning*, **45**(1), 5–32, (2001).
[8] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth, 1984.
[9] R. E. Bryant, 'Graph-based algorithms for Boolean function manipulation', *IEEE Transactions on Computers*, **C-35**(8), 677–692, (1986).
[10] S. Coste-Marquis and P. Marquis, 'On belief change for multi-label classifier encodings', in *Proc. of IJCAI'21*, pp. 1829–1836, (2021).
[11] M. Dalal, 'Investigations into a theory of knowledge base revision: Preliminary report', in *Proc. of AAAI'88*, pp. 475–479, (1988).
[12] A. Darwiche, 'SDD: A new canonical representation of propositional knowledge bases', in *Proc. of IJCAI'11*, pp. 819–826, (2011).
[13] A. Darwiche and P. Marquis, 'A knowledge compilation map', *Journal of Artificial Intelligence Research*, **17**, 229–264, (2002).
[14] A. de Colnet and P. Marquis, 'On translations between ML models for XAI purposes', in *Proc. of IJCAI'23*, (2023). To appear.
[15] 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).
[16] N. Gorji and S. Rubin, 'Sufficient reasons for classifier decisions in the presence of domain constraints', in *Proc. of AAAI'22*, pp. 5660–5667, (2022).
[17] L. Grinsztajn, E. Oyallon, and G. Varoquaux, 'Why do tree-based models still outperform deep learning on tabular data?', *CoRR*, **abs/2207.08815**, (2022).
[18] 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).
[19] D. Gunning, 'Darpa's explainable artificial intelligence (XAI) program', in *Proc. of IUI'19*, (2019).
[20] S.O. Hansson, 'Semi-revision (invited paper)', *J. Appl. Non Class. Logics*, **7**(2), (1997).
[21] X. Huang, Y. Izza, A. Ignatiev, and J. Marques-Silva, 'On efficiently explaining graph-based classifiers', in *Proc. of KR'21*, pp. 356–367, (2021).
[22] H. Katsuno and A. O. Mendelzon, 'On the difference between updating a knowledge base and revising it', in *Proc. of KR'91*, pp. 387–394, (1991).
[23] H. Katsuno and A. O. Mendelzon, 'Propositional knowledge base revision and minimal change', *Artificial Intelligence*, **52**(3), 263–294, (1991).
[24] S. Konieczny, M. Medina Grespan, and R. Pino Pérez, 'Taxonomy of improvement operators and the problem of minimal change', in *Proc. of KR'10*, pp. 161–170, (2010).
[25] F. Koriche, J.-M. Lagniez, P. Marquis, and S. Thomas, 'Knowledge compilation for model counting: Affine decision trees', in *Proc. of IJCAI'13*, pp. 947–953, (2013).
[26] J. Lang, P. Liberatore, and P. Marquis, 'Propositional independence: Formula-variable independence and forgetting', *Journal of Artificial Intelligence Research*, **18**, 391–443, (2003).
[27] 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).
[28] T. Miller, 'Explanation in artificial intelligence: Insights from the social sciences', *Artificial Intelligence*, **267**, 1–38, (2019).
[29] C. Molnar, *Interpretable Machine Learning*, Leanpub, 2020.
[30] L. Pulina and A. Tacchella, 'An abstraction-refinement approach to verification of artificial neural networks', in *Proc. of CAV'10*, pp. 243–257, (2010).
[31] J. Ross Quinlan, 'Induction of decision trees', *Machine Learning*, **1**(1), 81–106, (1986).
[32] 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).
[33] W. Samek, G. Montavon, A. Vedaldi, L.K. Hansen, and K.R. Müller, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, Springer, 2019.
[34] N. Schwind, K. Inoue, and P. Marquis, 'Editing boolean classifiers: A belief change perspective', in *Proc. of AAAI'23*, (2023).
[35] N. Schwind, S. Konieczny, and P. Marquis, 'On belief promotion', in *Proc. of KR'18*, pp. 297–307, (2018).
[36] R. Srinivasan and A. Chander, 'Explanation perspectives from the cognitive sciences - A survey', in *Proc. of IJCAI'20*, pp. 4812–4818, (2020).
[37] G. Van den Broeck, A. Lykov, M. Schleich, and D. Suciu, 'On the tractability of SHAP explanations', in *Proc. of AAAI'21*, pp. 6505–6513, (2021).
[38] I. Wegener, *Branching Programs and Binary Decision Diagrams*, SIAM, 2000.
[39] J. Yu, A. Ignatiev, P. J. Stuckey, N. Narodytska, and J. Marques-Silva, 'Eliminating the impossible, whatever remains must be true', *CoRR*, **abs/2206.09551**, (2022).
[40] Z.-H. Zhou, 'Abductive learning: towards bridging machine learning and logical reasoning', *Science China Information Science*, **62**(7), 76101:1–76101:3, (2019).