Rectifying Mono-Label Boolean Classifiers

Sylvie Coste-Marquis¹, Pierre Marquis¹,²
¹Univ. Artois, CNRS, CRIL
²Institut Universitaire de France
{coste, marquis}@cril.fr

Abstract
We elaborate on the notion of rectification of a Boolean classifier Σ introduced in [Coste-Marquis and Marquis, 2021]. Given Σ and some background knowledge T, postulates characterizing the way Σ must be changed into a new classifier Σ ∗ T that complies with T were presented. We focus here on the specific case of mono-label Boolean 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 ∗ satisfying the postulates, and (2) when Σ and T are Boolean circuits, we show how a classification circuit equivalent to Σ ∗ T can be computed in time linear in the size of Σ and T; when Σ and T are decision trees, a decision tree equivalent to Σ ∗ T can be computed in time polynomial in the size of Σ and T.

1 Introduction
Much work has been devoted for the past few years to eXplainable AI, in the objective of making ML models less opaque, see e.g., [Adadi and Berrada, 2018; Miller, 2019; Samek et al., 2019; Guidotti et al., 2019; Srinivasan and Chander, 2020; Molnar, 2020; Lundberg et al., 2020; Rudin et al., 2021]. Thus went typically through the definition of a number of explanation and/or verification issues for various ML models, and the development and the evaluation of algorithms for addressing those issues.

Verifying a model requires to be able to test whether the predictions made by the model are correct or not, and this often asks for leveraging the skills of an expert. Whenever the prediction made is viewed as incorrect or, more generally, when it conflicts with the expert knowledge, a more 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 expert knowledge.

To make things concrete, 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 our recent work [Coste-Marquis and Marquis, 2021] is relevant to Alice’s concern. In this work, we introduced a change operation, called rectification, that is suited to multi-label Boolean classifiers. Given a set X = {x₁, . . . , xₙ} (its elements are Boolean features) and a set Y = {y₁, . . . , yₘ}, that is disjoint with X (its elements are labels, denoting classes/concepts), X is the set {0, 1}ⁿ of all vectors over {0, 1} of size n, and 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 X to Y, associating with each input instance (a vector x ∈ X of n Boolean values assigned to the elements of X) a vector y ∈ Y of m Boolean values assigned to the elements of Y. Whenever an instance x = (x₁, . . . , xₙ) is associated by the classifier with y = (y₁, . . . , yₘ) such that yᵢ (i ∈ [m]) is equal to 1 (resp. 0), one considers that x belongs to the class yᵢ (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 = {x₁, x₂, x₃} and labels Y = {y}, associated with the following semantics:

- x₁ = 1: has low income
- x₂ = 1: has reimbursed a previous loan
- x₃ = 1: has debts
- y = 1: to grant the loan

Bob is viewed as the instance x = (x₁ = 1, x₂ = 1, x₃ = 0), and if f denotes the predictor used by Alice, we have f(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 \( \Sigma \) over a set of variables \( PS \) such that \( X \cup Y \subseteq PS \). In such a circuit \( \Sigma \), 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 \( \Sigma \) are also considered. They are represented as well by a Boolean circuit \( T \) over \( PS \).

The purpose of rectifying the classification circuit \( \Sigma \) by \( T \) is to modify \( \Sigma \) so that (1) the constraints imposed by \( T \) on the facts about \( Y \) that must hold under each \( x \) are respected, and (2) the resulting circuit noted \( \Sigma \ast T \) is still a classification circuit. A minimal change condition is taken into account; it states that the way \( x \) was classified by \( \Sigma \) must not be modified by the rectification process when the constraints imposed by \( T \) on the facts that hold under \( x \) are already satisfied. In the general case when \( Y \) is unconstrained, several classification circuits \( \Sigma \ast T \) can be found that satisfy (1) and the minimal change condition. Stated otherwise, several rectification operators \( \ast \) can be defined.

In [Coste-Marquis and Marquis, 2021], it was shown that a rectification operation amounts to a specific form of belief change [Alchourron et al., 1985]. A logical characterization of classification circuits has been pointed out and a number of postulates that rectification operators should satisfy have been presented. We 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, we proved that the families of rectification operators and those of “standard” belief change operators, namely revision operators / update operators [Katsuno and Mendelzon, 1991b; Katsuno and Mendelzon, 1991a], are disjoint.

In this paper, we focus on the specific case of mono-label Boolean 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 is the case for the credit scoring scenario). Under this assumption, we present two contributions. On the one hand, provided that the Boolean classifier is represented by a circuit \( \Sigma \) involving only variables \( X \) for the representation of instances and the concept variable \( y \), we show that there exists a unique rectification operator, noted \( \ast \), thus providing a full characterization of rectification operators in this restricted case. Since it is unique, \( \ast \) coincides with the operators presented in [Coste-Marquis and Marquis, 2021] when \( Y \) is a singleton. On the other hand, we show how a classification circuit equivalent to \( \Sigma \ast T \) can be computed in time linear in the size of \( \Sigma \) 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 [Coste-Marquis and Marquis, 2021], which are of size exponential in the size of \( X \). In addition, the specific cases when \( \Sigma \) and \( T \) are SDD circuits [Darwiche, 2011], OBDD circuits [Bryant, 1986], and (possibly affine) decision trees [Wegener, 2000; Koriche et al., 2013] are analyzed.

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. Section 5 concludes the paper.

2 Preliminaries

Before presenting the key definitions of the rectification setting pointed out in [Coste-Marquis and Marquis, 2021], we first need to recall a couple of notions about propositional representations.

A Boolean circuit \( \Phi \) over a set \( PS \) of propositional variables is a DAG where internal nodes are labelled by usual connectives, \( \neg, \lor, \land \), but may also correspond to decision nodes over variables from \( PS \), and leaves are labelled by variables from \( PS \) or by Boolean constants \( \top, \bot \). In addition, the specific cases when \( \Phi \) is a singleton. On the other hand, we show how a classification circuit equivalent to \( \Sigma \ast T \) can be computed in time linear in the size of \( \Sigma \) 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 [Coste-Marquis and Marquis, 2021], which are of size exponential in the size of \( X \). In addition, the specific cases when \( \Sigma \) and \( T \) are SDD circuits [Darwiche, 2011], OBDD circuits [Bryant, 1986], and (possibly affine) decision trees [Wegener, 2000] [Koriche et al., 2013] are analyzed.

A formula over \( PS \) is simply 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 \( X \) of \( PS \), \( \mathcal{L}_X \) 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 consistent term into which every variable of \( X \) occurs (as such, or negated). In the following, every instance \( x \in X \) is also viewed as a canonical term over \( X \), still noted \( x \), such that for every \( i \in [n] \), \( x_i \) is a literal of this term if the \( i^{th} \) coordinate of \( x \) 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 \( \omega \) from \( V \) to \( B = \{0, 1\} \). Every interpretation over \( V \) corresponds to a unique canonical term over \( V \), and vice versa. When a total ordering \( \prec \) over \( PS \) is provided, interpretations can be represented by bit vectors or by words. For instance, if \( V = \{ v_1, v_2 \} \) such that \( v_1 \prec v_2 \), then the mapping \( \omega \) 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 $V = \text{Var}(\Phi)$, we use $\omega \models \Phi$ to denote the fact that $\omega$ if a model of $\Phi$ according to the semantics of propositional logic. That is, assigning the variables of $\Phi$ to truth values as specified by $\omega$ makes $\Phi$ true. By $[\Phi]$ we denote the set of models of $\Phi$ over $\text{Var}(\Phi)$. $\Phi$ is inconsistent if $[\Phi] = \emptyset$, and $\Phi$ is consistent otherwise. A Boolean circuit $\Psi \in L_{\Gamma}$ is a logical consequence of a Boolean circuit $\Phi \in L_{\Gamma}$ (denoted $\Phi \models \Psi$) if $\Phi \wedge \neg \Psi$ is inconsistent. $\Phi$ and $\Psi$ are logically equivalent (denoted $\Phi \equiv \Psi$) if they have the same models over $\text{Var}(\Phi) \cup \text{Var}(\Psi)$.

Given a Boolean circuit $\Phi \in L_{\Gamma}$ and a consistent term $\gamma$ over $PS_{\Gamma}$, the conditioning of $\Phi$ by $\gamma$ is the Boolean circuit from $L_{\Gamma}$, noted $\Phi(\gamma)$, obtained by replacing in $\Phi$ every occurrence of a variable $v \in \text{Var}(\gamma)$ by $\top$ if $v$ is a positive literal of $\gamma$ and by $\bot$ if $\neg \gamma$ is a negative literal of $\gamma$.

When $V$ is a subset of $PS_{\Gamma}$, a Boolean circuit $\Phi \in L_{\Gamma}$ is said to be independent of $V$ if there is a Boolean circuit $\Psi \in L_{\Gamma}$ logically equivalent to $\Phi$ such that $\text{Var}(\Psi) \cap V = \emptyset$. The forgetting of $\Phi$ in $\Psi$, denoted $\Psi \Phi$, is (up to logical equivalence) the strongest logical consequence of $\Phi$ that is independent of $V$ (see e.g., [Lang et al., 2003]). The projection of $\Sigma$ onto $V$ is the forgetting of $\Phi$ in $\Sigma$, where $\Sigma$ denotes the set $PS \setminus V$. $\exists V \Phi$ can be characterized as follows:

- $\exists ! \Phi \equiv \Phi$,
- $\exists \{v\} \Phi \equiv (\Phi |_{\overline{v}}) \lor (\Phi(v))$,
- $\exists \{v' \cup \{v\}\} \Phi \equiv \exists V'. (\exists \{v\}. \Phi)$.

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

The last notion to be made precise before defining rectification operators is the notion of fact compliance. A Boolean circuit $\Sigma \in L_{\Gamma}$ is fact-compliant with a Boolean circuit $T \in L_{\Gamma}$ on an instance $x$ if and only if $\Sigma(x) \models F(T, x)$ where $F(T, x) = \top$ if $T(x)$ is inconsistent, $= \land_{t \in L_{\Gamma}} s.t. \ T(x_t) = \top$ otherwise.

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

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

Now, let $T = ((x_1 \land x_2) \Rightarrow (y_1 \land y_2)) \land ((x_1 \land \overline{y_2}) \Rightarrow (y_1 \lor \overline{y_2})) \land ((\overline{x_1} \land x_2) \Rightarrow \overline{y_1} \land x_1 \lor x_2)$. $T$ 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 $F(T, x_1 \land x_2) \equiv y_1 \land y_2$, $F(T, x_1 \land \overline{y_2}) \equiv \top$, $F(T, \overline{x_1} \land x_2) \equiv \overline{y_1}$, and $F(T, \overline{x_1} \land \overline{y_2}) \equiv \top$. Thus, $\Sigma$ 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 $\ast$ is a mapping associating with two given circuits $T$ and $\Sigma$ from $L_{\Gamma}$, where $\Sigma$ has the $XY$-classification property, a circuit from $L_{\Gamma}$, noted $\Sigma \ast T$ and called a rectified circuit, such that:

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

The rationale for those postulates is presented in [Coste-Marquis and Marquis, 2021]. 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 $x$ as achieved by $\Sigma$ should not be modified by the rectification operation whenever $\Sigma$ is fact-compliant with $T$ on $x$. (RE3) is a success condition: it demands that the rectified circuit $\Sigma \ast T$ is fact-compliant with $T$ on every $x$. (RE4) is a non-triviality condition; it deals with the case when $T$ is inconsistent; in such a situation, a minimal change of $\Sigma$ 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 $\Sigma$ by $T$ must not depend on the variables outside $X \cup Y$.

**Example 2.** Let us consider again the classification circuit $\Sigma$ and the formula $T$ presented at Example [1]. Since $\Sigma$ is fact-compliant with $T$ on every instance, but $(0, 1)$, (RE2) imposes that $(\Sigma \ast T)((1, 1)) \equiv y_1 \land y_2$, $(\Sigma \ast T)((1, 0)) \equiv y_1 \land \overline{y_2}$, and $(\Sigma \ast T)((0, 0)) \equiv \overline{y_1} \land \overline{y_2}$. (RE3) requires that $(\Sigma \ast T)((1, 0)) \equiv \overline{y_1}$. Finally, (RE1) ensures that $(\Sigma \ast T)((0, 1)) \equiv \overline{y_1} \land \overline{y_2}$ or $(\Sigma \ast T)((0, 0)) \equiv y_1 \land y_2$. Accordingly, the classification $\overline{y_1} \land y_2$ of the instance $(0, 1)$ as achieved by $\Sigma$ 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 $\overline{y_1} \land y_2$ by $\overline{y_1}$ may lead to $y_1 \land \overline{y_2}$ 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 $\ast$ satisfying the rectification postulates in the restricted case when $Y$ contains a single label:

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

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

**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 ($\overline{y_1}$) but has not reimbursed a previous loan ($\overline{y_2}$), or (which looks more risky) a customer who has low incomes ($x_1$) and has some debts ($x_2$). 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 $\Sigma_X = (\overline{y_1} \land y_2) \lor (x_1 \land x_2)$. Alice’s expertise can be represented by the formula $T = ((x_1 \land \overline{y_2}) \Rightarrow y) \land (\overline{x_2} \Rightarrow \overline{y})$ encoding her two decision rules. For every instance $x \in X$, Table 1 indicates from left to right, whether or not $\Sigma$ classifies $x$ as positive (this is the case precisely when $\Sigma(x) \equiv y$), the constraint imposed by $T$ on the way $x$ should be classified (i.e., as positive when $T(x) \equiv y$ and as negative when $T(x) \equiv \overline{y}$, the facts $F(T, x)$ that hold in $T$ under $x$, and finally whether or not $\Sigma \ast T$ classifies $x$ as positive (this is the case precisely when $(\Sigma \ast T)(x) \equiv y$).

| $x$ | $\Sigma(x)$ | $T(x)$ | $F(T, x)$ | $(\Sigma \ast T)(x)$ |
|-----|-------------|--------|-----------|---------------------|
| 000 | $y$         | $\overline{y}$ | $y$       | $\overline{y}$       |
| 001 | $y$         | $\overline{y}$ | $\overline{y}$ | $\overline{y}$       |
| 100 | $\overline{y}$ | $\top$ | $\top$ | $\top$ |
| 101 | $\overline{y}$ | $\top$ | $\top$ | $\top$ |
| 110 | $\overline{y}$ | $y$ | $y$ | $y$ |
| 111 | $\top$ | $\top$ | $\top$ | $\top$ |

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

Note that when $Y$ is a singleton, for every $x \in X$ such that $T(x)$ is consistent, we have $T(x) \equiv F(T, x)$. Then (RE3) shows immediately that for every $x \in X$ such that $T(x)$ is consistent, $\Sigma \ast T$ is knowledge-compliant with $T$ on $x$, i.e., $(\Sigma \ast T)(x) \equiv T(x)$ [Coste-Marquis and Marquis, 2021].

4 Representing Rectified Classifiers

Some rectification operators have been put forward in [Coste-Marquis and Marquis, 2021]. Among them is the following $\ast_D$ operator:

**Definition 2 ($\ast_D$).** Let $\odot_D$ denote Dalal revision operator [Dalal, 1988]. Let $\ast_D$ be the mapping associating with $T \in$ 1Given two propositional representations $\varphi$ and $\alpha$, the models of $\varphi \odot_D \alpha$ consist of the models of $\alpha$ which are as close as possible to $\varphi$ w.r.t. Hamming distance.
\( \mathcal{L} \) and a classification circuit \( \Sigma \in \mathcal{L} \), a classification circuit \( \Sigma \star D \ T \in \mathcal{L} \) such that

\[
\Sigma \star D \ T \equiv \bigvee_{x \in X} x \land (\Sigma \star D \ T)(x)
\]

where for any \( x \in X \), \((\Sigma \star D \ T)(x) = \Sigma(x) \circ D(F(T,x)) \).

It was already observed that \( \star_D \) coincides with other rectification operators pointed out in [Coste-Marquis and Marquis, 2021] 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 \( \bigvee_{x \in X} x \land (\Sigma \star D \ T)(x) \) 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 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 \( \Sigma_T \) from \( \mathcal{L}_X \) so that \( \Sigma \star T = \Sigma_T \Leftrightarrow y \). Thus, generating a circuit representing \( \Sigma \star T \) boils down to generating a circuit representing \( \Sigma_T \).

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 \( x \in X \) and \( T \in \mathcal{L} \). \( T \) classifies \( x \) as

- a positive instance if \( x \models T(y) \land \neg T(\overline{y}) \);
- a negative instance if \( x \models T(\overline{y}) \land \neg T(y) \).

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

On this basis, the following representation of \( \Sigma_T \) can be derived:

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

\[
\Sigma_T \equiv (\Sigma_X \land \neg (T(\overline{y}) \land \neg T(y))) \lor (T(y) \land \neg T(\overline{y})).
\]

**Proof.** The result comes directly from the identification of the only two reasons according to which an instance \( x \in X \) must be classified as positive by the rectified classifier (i.e., it must be a model of \( \Sigma_X \);

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

The rationale of this characterization of \( \Sigma_T \) is as follows. For an instance \( x \) 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 \( T(y) \land \neg T(\overline{y}) \)), or that the classification circuit considered at start classifies \( x \) as positive, provided that \( T \) does not consistently ask \( x \) to be classified as negative (this corresponds to the disjunct \( \Sigma_X \land \neg (T(\overline{y}) \land \neg T(y)) \)). Such a construction is reminiscent to the representation of STRIPS-like actions using propositional formulas, 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, \( \Sigma_T \Leftrightarrow y \) (where \( \Sigma_T \) 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 \( \Sigma \) and \( T \) as in Example 3. We have \( T(y) \equiv x_2 \land T(\overline{y}) \equiv \overline{x_2} \lor x_3 \). Thus, we get \( \Sigma_T \equiv (x_1 \land x_3) \land \neg (x_1 \lor x_3) \). This circuit can be simplified as \( \Sigma_T \equiv x_1 \land x_2 \). One can check in Table 1(rightmost column) that the models of \( \Sigma_T \) are precisely those \( x \in X \) such that \( (\Sigma \star T)(x) = y \). 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.

Note that if \( \Sigma_X \) and \( T \) are formulae (and not Boolean circuits) in Proposition 3 then the resulting characterization of \( \Sigma_T \) also is a formula (indeed, conditioning a formula leads to a formula). Furthermore, whenever \( \Sigma_X \) and \( T \) belongs to a class \( C \) of circuits that offers in polynomial time the transformations of negation (\( \neg C \)), bounded conjunction (\( \land BC \)), and bounded disjunction (\( \lor BC \)) [Darwiche and Marquis, 2002], a representation of \( \Sigma_T \) in \( C \) can be derived in polynomial time from \( \Sigma_X \) and \( T \).

Notably, focusing on a restricted class of circuits \( C \) is not mandatory for ensuring tractable classification: when \( \Sigma_T \) is in \( \mathcal{L}_X \), deciding whether or not \( x \in X \) is classified as positive by \( \Sigma \land T \) amounts to testing whether or not \( x \) is a model of \( \Sigma_T \), 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 prove valuable for other reasons, especially from an eXplainable AI perspective (see e.g., [Audemard et al., 2020], [Barcelós et al., 2020], [Arenas et al., 2021], [Van den Broeck et al., 2021], [Huang et al., 2021]). Among the classes of circuits offering ¬C, ∧BC, and ∨BC are SDD, the class of sentential decision diagrams [Darwiche, 2011], OBDD, the class of ordered binary decision diagrams [Bryant, 1986], but also DT, the class of decision trees, and more generally ADT, the class of affine decision trees [Koriche et al., 2013]. 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 classification rule like (x1 ∧ ¬x3) ⇒ y (equivalent to the clause ¬x1 ∨ x3 ∨ y) that is entailed by the formula T considered in Example 4 could be easily handled.

The case of DT is of particular interest since it corresponds to a well-known ML model [Breiman et al., 1984], [Quinlan, 1986], that also serves as a key component of other ML models, especially random forests RF [Breiman, 2001] and boosted trees [Wegener, 2000]. Furthermore, DT is a much more intelligible model than most of Boolean classifiers [Audemard et al., 2021], and RF also offers some tractable explanation facilities (via the notion of majority reason) [Audemard et al., 2022]. Given the significance of those ML models, the existence of polynomial-time algorithms for rectifying decision trees and random forest is a noteworthy consequence of Proposition 3.

Example 5. Considering Example 4 again, let us finally illustrate how a decision tree classifier equivalent to Σ ∗ T can be generated in polynomial time from Σ and T. Starting with a decision tree Σ over X ∪ {y}, a decision tree over X equivalent to ΣX (given at Figure 2) can be obtained by conditioning Σ by y since ΣX ∼= Σ(y). Conditioning a decision tree by a literal v (resp. ¬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 ∪ {y} at Figure 4, one can derive efficiently decision trees for T(∀) and T(y) (in Figure 4 they are the subtrees rooted at nodes T(∀) and T(y)).

On this ground, deriving decision trees equivalent to T(∀) ∧ ¬T(y) and T(y) ∧ ¬T(∀), as reported on Figures 5 and 6 (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 [Wegener, 2000]. 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 6 presents a decision tree equivalent to ΣX ∧ ¬(T(∀) ∧ ¬T(y)) and obtained by conjoining the decision tree of ΣX given in Figure 2 (left) with the negation of the decision tree of T(∀) ∧ ¬T(y) given in Figure 4. Figure 6 illustrates the effect of the simplification process.

Figure 7 presents a decision tree equivalent to (ΣX ∧ ¬(T(∀) ∧ ¬T(y))) ∨ (T(y) ∧ ¬T(∀)), obtained by disjoining the decision tree at Figure 4 with the decision tree at Figure 5. 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 7 presents an equivalent, yet simplified decision tree (obtained by running the simplification procedure sketched above). By construction, it is equivalent to ΣX. As expected, one recovers here the condition x_1 ∧ 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 Σ ∗ T can be generated in linear time from ΣX 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.

Rectifying a random forest or an (AdaBoost-style) boosted tree simply amounts to rectifying every decision tree in it.
5 Conclusion

The main contribution of this paper is twofold. On the one hand, we have presented a characterization theorem for the unique rectification operator \( \star \) obtained when dealing with mono-label Boolean classifiers. On the other hand, we have explained how a classification circuit equivalent to \( \Sigma \star T \) can be computed in time linear in the size of \( \Sigma \) and \( T \). Especially, we have also shown that a decision tree equivalent to \( \Sigma \star T \) can be computed in time polynomial in the size of \( \Sigma \) and \( T \) when each of \( \Sigma \) and \( T \) is represented as a decision tree.

In eXplainable AI, a contrastive explanation for an instance \( x \) aims to explain why \( x \) has not been classified by the ML model as the explainee expected it (thus, addressing the “Why not?” question) [Miller, 2019; Ignatiev et al., 2020]. When the explainee is not only surprised by the prediction made by the classifier, but actually believes that the prediction is wrong, pointing out an explanation is not enough. A rectification process must take place. We have shown how to achieve this process efficiently when the mono-label Boolean classifier at hand is a decision tree or a classifier based on such trees (random forests, boosted trees).

In this work, one started with the basic assumption that the available background knowledge \( T \) is more reliable than the classification circuit \( \Sigma \). 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 [Hansson, 1997], promotion [Schwind et al., 2018], or improvement [Konieczny et al., 2010], 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. This is left for further research.

Acknowledgements

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

[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.

[Alchourrón et al., 1985] 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.

[Arenas et al., 2021] 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*, pages 6670–6678, 2021.
Audemard et al., 2020] G. Audemard, F. Koriche, and P. Marquis. On tractable XAI queries based on compiled representations. In Proc. of KR’20, pages 838–849, 2020.

Audemard et al., 2021] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Laglize, and P. Marquis. On the computationally intelligibility of Boolean classifiers. In Proc. of KR’21, pages 74–86, 2021.

Audemard et al., 2022] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Laglize, and P. Marquis. Trading complexity for sparsity in random forest explanations. In Proc. of AAAI’22, 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.

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

Coste-Marquis and Marquis, 2021] S. Coste-Marquis and P. Marquis. On belief change for multi-label classifier encodings. In Proc. of IJCAI’21, pages 1829–1836, 2021.

Dalal, 1988] M. Dalal. Investigations into a theory of knowledge base revision: Preliminary report. In Proc. of AAAI’88, pages 475–479, 1988.

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

Darwiche, 2011] A. Darwiche. SDD: A new canonical representation of propositional knowledge bases. In Proc. of IJCAI’11, pages 819–826, 2011.

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.

Hansson, 1997] S.O. Hansson. Semi-revision (invited paper). J. Appl. Non Class. Logics, 7(2), 1997.

Huang et al., 2021] Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, and João Marques-Silva. On efficiently explaining graph-based classifiers. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, Proc. of KR’21, pages 356–367, 2021.

Ignatiev et al., 2020] A. Ignatiev, N. Narodytska, N. Asher, and J. Marques-Silva. On relating ‘why?’ and ‘why not?’ explanations. CoRR, abs/2012.11067, 2020.

Katsuno and Mendelzon, 1991a] H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. Artificial Intelligence, 52(3):263–294, 1991.

Konieczny et al., 2010] S. Konieczny, M. Medina Grespan, and R. Pino Pérez. Taxonomy of improvement operators and the problem of minimal change. In Proc. of KR’10, pages 161–170, 2010.

Koriche et al., 2013] F. Koriche, J.-M. Laglize, P. Marquis, and S. Thomas. Knowledge compilation for model counting: Affine decision trees. In Proc. of IJCAI’13, pages 947–953, 2013.

Lang et al., 2003] J. Lang, P. Liberatore, and P. Marquis. Propositional independence: Formula-variable independence and forgetting. Journal of Artificial Intelligence Research, 18:391–443, 2003.

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.

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

Molnar, 2020] C. Molnar. Interpretable Machine Learning. Leanpub, 2020.

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

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.

Schwind et al., 2018] N. Schwind, S. Konieczny, and P. Marquis. On belief promotion. In Proc. of KR’18, pages 297–307, 2018.

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.

Van den Broeck et al., 2021] G. Van den Broeck, A. Lykov, M. Schleich, and D. Suciu. On the tractability of SHAP explanations. In Proc. of AAAI’21, pages 6505–6513, 2021.

Wegener, 2000] I. Wegener. Branching Programs and Binary Decision Diagrams. SIAM, 2000.