Rectifier pour mieux distiller

Gilles Audemard*, Sylvie Coste-Marquis*, Pierre Marquis*,**, Mehdi Sabiri*,
Nicolas Szczepanski***

*Univ. Artois, CNRS, CRIL
**Institut Universitaire de France
***IRT SystemX

Résumé. Nous présentons dans cet article une approche de distillation d'arbres optimisés en arbres de décision. Une telle distillation vise à aboutir à un modèle offrant un compromis acceptable en terme de précision et d'interprétabilité. Nous expliquons comment l'approche de correction de classeurs binaires, appelée rectification, introduite récemment peut être utilisée pour mettre en œuvre un tel processus de distillation. Nous montrons empiriquement qu'elle fournit des résultats intéressants, en comparaison de la distillation par ré-entraînement.

1 Introduction

L'interprétabilité d'un modèle d'apprentissage automatique (ML) peut être considérée selon plusieurs points de vue. Ceux-ci incluent la clarté de la méthode utilisée pour construire le modèle, le nombre de paramètres, la structure ou la taille du modèle, la possibilité d'en extraire des règles de classement simples ou des explications (voir par exemple (Molnar, 2019)). Ainsi, les arbres de décision constituent un modèle ML plutôt considéré comme interprétable (quoique souvent peu précis). En particulier, nous avons montré dans des travaux passés que les arbres de décision supportent en temps polynomial une large palette de requêtes d'explication ou de vérification, dont les réponses peuvent être exploitées par l'utilisateur pour décider de faire confiance ou pas au modèle ML qui a été appris et/ou aux classements qui en sont issus (Audemard et al., 2021, 2022a). En pratique, les réponses aux requêtes posées sont dérivables en un temps raisonnable, même quand l'arbre considéré en entrée contient beaucoup de nœuds et n'est donc pas interprétable globalement. A contrario, les réseaux de neurones profonds ou les arbres optimisés (*boosted trees*) sont en général précis mais peu interprétables, même au sens de l'ensemble de requêtes XAI offertes en temps polynomial.

La *distillation* de modèle (Hinton et al., 2015; Frosst et Hinton, 2017; Gou et al., 2021) est une méthode de construction d'un modèle ML cible « plus simple » que le modèle ML source considéré en entrée ; le modèle source et le modèle cible peuvent être de la même famille, mais pas forcément. La simplicité recherchée peut s'exprimer en terme de nombre de paramètres, de valeurs d'hyperparamètres et viser diverses finalités, comme diminuer la quantité de mémoire requise pour faire fonctionner le modèle, réduire les temps requis pour obtenir des prédictions, fournir des explications (Asadulaev et al., 2019; Wood-Doughty et al., 2022), etc. La distillation constitue, en particulier, une approche possible pour aboutir à un bon

compromis précision / interprétabilité, en permettant de construire un modèle interprétable I suffisamment précis à partir d'un modèle P peu interprétable mais précis.

Dans cet article, on se focalise sur un processus de distillation incrémentale, où il s'agit de corriger un modèle ML I de départ, plutôt interprétable mais peu précis, en utilisant un modèle ML P, plutôt précis mais peu interprétable, et considéré comme oracle. I et P sont ici deux classeurs binaires sur un ensemble X de n attributs booléens (vus également comme des variables propositionnelles). Formellement, de tels classeurs sont des applications de l'espace $\boldsymbol{X} = \{0,1\}^n$ des instances dans $\{0,1\}$. Dans la suite, la famille de modèles I considérée est celle des arbres de décision (Breiman et al., 1984; Quinlan, 1986), celle des modèles P est la famille des arbres optimisés (*boosted trees*) (Freund et Schapire, 1997).

L'approche standard utilisée à des fins de correction en apprentissage automatique est le *ré-entraînement* : quand on rencontre une instance $\boldsymbol{x}$ telle que $I(\boldsymbol{x}) \neq P(\boldsymbol{x})$, on suppose que l'oracle P a raison. On ajoute $(\boldsymbol{x}, P(\boldsymbol{x}))$ à l'ensemble d'apprentissage et on ré-apprend I.

La principale contribution de cet article est de montrer comment l'opération de correction de classeurs, appelée *rectification*, introduite dans (Coste-Marquis et Marquis, 2021), *peut être utilisée avec profit à des fins de distillation*. En toute généralité, l'opération de rectification prend en entrée un *circuit de classement* Σ et une formule T représentant des connaissances expertes, plus fiables que celles déductibles de Σ[1]. Σ et T sont construits sur les variables propositionnelles de $X \cup \{y\}$. Les éléments de X correspondent aux conditions booléennes utilisées pour décrire les instances et y désigne la variable de classe. Un circuit de classement sur $X \cup \{y\}$ est un circuit équivalent à une formule de la forme $\Sigma_X \Leftrightarrow y$ où Σ_X est une formule propositionnelle sur X. Tout classeur binaire C s'appuyant sur des attributs X booléens peut être vu comme une formule propositionnelle sur X (notée également C pour éviter un formalisme trop pesant) dont les modèles sont précisément les instances classées positivement par C, i.e., vérifiant $C(\boldsymbol{x}) = 1$. À un tel C, on peut associer le circuit de classement $\Sigma = C \Leftrightarrow y$. Ce circuit Σ permet de classer tout $\boldsymbol{x} \in \boldsymbol{X}$: $\boldsymbol{x}$ est classé positivement (resp. négativement) par Σ quand Σ conditionnée par le terme canonique $t_{\boldsymbol{x}}$ associé à $\boldsymbol{x}$ (i.e., le terme qui décrit totalement $\boldsymbol{x}$) équivaut à y (resp. $\overline{y}$). Les connaissances fournies par T peuvent impliquer des *règles de classement* de la forme $\varphi_X \Rightarrow y$ (une règle sur y) ou $\varphi_X \Rightarrow \overline{y}$ (une règle sur $\overline{y}$) où φ_X est une formule sur X. Toutefois, aucune condition n'est imposée sur T hormis d'être construite sur $X \cup \{y\}$; ainsi, les règles déductibles de T quand il en existe peuvent être *conflictuelles* (on peut avoir deux règles déductibles de T qui ont des conclusions contradictoires – y pour l'une et $\overline{y}$ pour l'autre – et des prémisses non contradictoires) et *incomplètes* (il se peut qu'aucune règle de classement déductible de T ne couvre une instance $\boldsymbol{x}$ donnée).

Dans le cas du classement binaire, il a été montré récemment (Coste-Marquis et Marquis, 2023) qu'il existe *un opérateur de rectification unique*. La rectification $\Sigma \star T$ de Σ par T désigne un circuit de classement obtenu en modifiant Σ minimalement, de sorte que pour toute instance $\boldsymbol{x}$ qui n'est pas couverte par des règles de classement conflictuelles déductibles de T, $\boldsymbol{x}$ est classée par $\Sigma \star T$ comme demandé par T, et pour toutes les autres instances, $\boldsymbol{x}$ est classée par $\Sigma \star T$ comme elle est classée par Σ.

Dans le travail décrit ici, nous mettons en place une approche de *distillation incrémentale* d'arbre de décision I en utilisant un arbre optimisé P pour oracle. Nous montrons d'abord qu'il est possible d'opérer de façon itérative le processus de rectification d'un circuit de classement $I \Leftrightarrow y$ par un ensemble de règles de classement déduites de $P \Leftrightarrow y$. Cela signifie précisément

1. Pour une approche combinant apprentissage et raisonnement à des fins de correction, voir aussi (Zhou, 2019).

que la rectification itérée d'un circuit de classement par une suite de telles règles équivaut à sa rectification en une seule étape par la conjonction des règles considérées (cette propriété n'est pas vérifiée en général quand la formule avec laquelle on rectifie est quelconque). Nous montrons ensuite comment construire des règles de classement déductibles de $P \Leftrightarrow y$ et utiles à la correction de $I \Leftrightarrow y$. Le principe est le suivant : pour chaque instance rencontrée $\boldsymbol{x}$ telle que $I(\boldsymbol{x}) \neq P(\boldsymbol{x})$, nous calculons une explication abductive t pour $\boldsymbol{x}$ étant donné P (Ignatiev et al., 2019) en utilisant l'approche présentée dans (Audemard et al., 2023) et nous fabriquons une règle de classement $R = t \Rightarrow y$ quand $P(\boldsymbol{x}) = 1$ et une règle de classement $R = t \Rightarrow \overline{y}$ quand $P(\boldsymbol{x}) = 0$. Par construction, cette règle de classement est valide pour P dans le sens où toute instance $\boldsymbol{x}'$ couverte par t est nécessairement classée par P de la même manière que $\boldsymbol{x}$. Puis nous corrigeons I en procédant d'une part par ré-entraînement (on ajoute à l'ensemble d'apprentissage un échantillon d'instances $(\boldsymbol{x}', P(\boldsymbol{x}))$ telles que t couvre $\boldsymbol{x}'$ et on assure que $(\boldsymbol{x}, P(\boldsymbol{x}))$ fait partie des instances ajoutées) et d'autre part en rectifiant le circuit de classement $I \Leftrightarrow y$ par R. Quand R est une règle de classement, la transformation XAI qui consiste à rectifier $I \Leftrightarrow y$ par R pour produire un circuit de classement $I^R \Leftrightarrow y$ où I^R est un arbre de décision, est réalisable en temps polynomial dans la taille de I plus la taille de R (Coste-Marquis et Marquis, 2023). Une autre contribution de ce travail est une évaluation expérimentale de la qualité de la distillation produite, réalisée en mesurant d'une part la précision empirique de l'arbre de décision corrigé I^R relativement à P, mais aussi sa taille. Sur cette base, nous pouvons comparer les deux méthodes de correction employées. Les résultats expérimentaux que nous avons obtenus montrent l'intérêt de l'approche de distillation par rectification. Pour des raisons d'espace, les preuves des propositions présentées dans le papier sont accessibles seulement en ligne, à partir du site `www.cril.univ-artois.fr/expekctation/`

2 Préliminaires formels

Soit $X = \{x_1, \ldots, x_n\}$ un ensemble de variables booléennes et soit y une variable booléenne n'apparaissant pas dans X, servant à décrire les classes. X correspond à l'ensemble des conditions booléennes apparaissant dans P et peut donc être vu comme un ensemble d'attributs booléens. Ces attributs servent à décrire les instances dont l'ensemble est noté $\boldsymbol{X}$. Les éléments de X ne représentent pas nécessairement des conditions indépendantes car ces conditions peuvent être issues des mêmes attributs numériques ou catégoriels (par exemple, on peut trouver dans P la condition $x_1 = (S > 30)$ portant sur l'attribut numérique S mais aussi la condition $x_2 = (S > 20)$ qui lui est liée logiquement : x_1 ne peut pas être vrai alors que x_2 serait faux). Une théorie du domaine, sous forme d'une formule logique sur X, précise les liens entre conditions booléennes non indépendantes (par exemple, $x_1 \Rightarrow x_2$).

Chaque instance $\boldsymbol{x}$ de $\boldsymbol{X}$ peut être considérée comme une interprétation sur X, associant la variable x_i ($i \in [n]$) à 1 si et seulement si la ième coordonnée $\boldsymbol{x}_i$ de l'instance $\boldsymbol{x}$ vaut 1 et elle peut être représentée par un terme canonique $t_{\boldsymbol{x}}$ sur X, formé par l'ensemble (conjonctivement interprété) des littéraux positifs x_i ($i \in [n]$) tels que $\boldsymbol{x}_i = 1$ et des littéraux négatifs $\overline{x_i}$ ($i \in [n]$) tels que $\boldsymbol{x}_i = 0$.

Quand T est une formule propositionnelle (ou, plus généralement, un circuit) sur $X \cup \{y\}$ et z est une variable de $X \cup \{y\}$, on note $T(z)$ (resp. $T(\overline{z})$) le conditionnement de T par z (resp. par $\overline{z}$). $T(z)$ (resp. $T(\overline{z})$) est la formule (ou le circuit) obtenu(e) en remplaçant dans T

toute occurrence de z par la constante booléenne $\top$ (resp. $\bot$). Ainsi, quand $\Sigma = \Sigma_X \Leftrightarrow y$ est un circuit de classement sur $X \cup \{y\}$, $\Sigma(y)$ a précisément pour modèles les modèles de Σ_X. Quand $\boldsymbol{x}$ est une instance, $T(\boldsymbol{x})$ désigne le conditionnement (itéré) de T par chaque littéral de $t_{\boldsymbol{x}}$. Par construction, $T(\boldsymbol{x})$ équivaut toujours à l'une des formules suivantes $\top$, $\bot$, y ou $\overline{y}$ (et à une des deux formules y ou $\overline{y}$ chaque fois que T est un circuit de classement sur $X \cup \{y\}$).

Les arbres de décision que nous considérons et les arbres optimisés sont des représentations de classeurs binaires :

Définition 1 *Un* arbre de décision (resp. un arbre de régression) *I sur X est un arbre binaire dont les nœuds internes sont étiquetés par des éléments de X et les feuilles par des constantes booléennes (resp. des nombres réels). Une instance $\boldsymbol{x} \in \boldsymbol{X}$ vérifie $I(\boldsymbol{x}) = v$ si et seulement si l'unique chemin de la racine de I à une feuille de I, qui est compatible avec $\boldsymbol{x}$, conduit à une feuille étiquetée par v.*

Définition 2 *Un* arbre optimisé *P sur X est un ensemble $\{A_1, \ldots, A_m\}$ d'arbres de régression sur X. Une instance $\boldsymbol{x} \in \boldsymbol{X}$ vérifie $P(\boldsymbol{x}) = 1$ si et seulement si $\sum_{i=1}^{m} A_j(\boldsymbol{x}) > 0$.*

Exemple 1 *À titre d'illustration, considérons un problème d'attribution de crédits à des clients d'une banque, caractérisés par un salaire annuel S, le fait d'avoir déjà remboursé un crédit REM et de disposer ou pas d'un CDI. Deux prédicteurs, P, un arbre optimisé comportant deux arbres de régression (A_1 et A_2), et I, un arbre de décision, ont été appris. Il sont décrits à la figure 1. X correspond aux quatre conditions booléennes considérées dans cet ordre : $x_1 = (S > 30)$, $x_2 = (S > 20)$, $x_3 = REM$, $x_4 = CDI$. x_1 et x_2 ne sont pas indépendantes. On peut vérifier que I et P classent les instances de $\boldsymbol{X}$ de la même manière, sauf les instances $(0, 1, 1, 1)$, $(0, 1, 0, 1)$ et $(1, 1, 1, 0)$. En effet, ces trois instances sont classées positivement par I et négativement par P.*

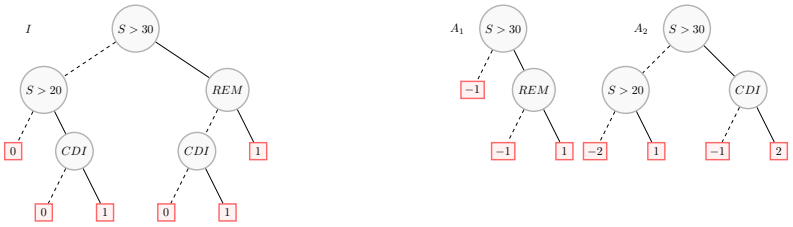


FIG. 1 – Un arbre de décision I et un arbre optimisé P, formé de deux arbres de régression A_1 et A_2. Pour chaque arbre, l'arc en pointillés (resp. l'arc en trait plein) issu d'un nœud étiqueté par une condition c correspond à l'affectation où c est fausse (resp. vraie). Dans un souci de clarté, les conditions sont exprimées en utilisant les attributs primitifs S, REM et CDI utilisés pour apprendre P.

Une explication abductive pour une instance $\boldsymbol{x}$ étant donné un classeur binaire C fournit un ensemble de conditions (des littéraux) qui explique pourquoi l'instance $\boldsymbol{x}$ a été classée par C de la façon dont elle a été classée.

Définition 3 *Une* explication abductive *pour $\boldsymbol{x} \in \boldsymbol{X}$ étant donné un classeur binaire C est un terme t sur X tel que t couvre $\boldsymbol{x}$ (i.e., $t \subseteq t_{\boldsymbol{x}}$) et pour tout $\boldsymbol{x}' \in \boldsymbol{X}$ tel que t couvre $\boldsymbol{x}'$, on a $C(\boldsymbol{x}') = C(\boldsymbol{x})$.*

Exemple 2 (suite de l'exemple 1) $t = \overline{x_1} = \overline{(S > 30)}$ *est une explication abductive pour* $(0, 1, 1, 1)$ *étant donné* P*. Le fait que le client dispose d'un salaire annuel inférieur (ou égal) à 30k€ par an suffit à expliquer pourquoi, selon* P*, le prêt demandé ne doit pas lui être accordé.*

3 Rectifier un circuit de classement par des règles déduites d'un circuit de classement

En toute généralité, lorsqu'un circuit de classement $\Sigma = \Sigma_X \Leftrightarrow y$ est rectifié par une formule quelconque T sur $X \cup \{y\}$, le résultat de la rectification est le circuit de classement $\Sigma \star T$ défini par $\Sigma \star T = \Sigma_X^T \Leftrightarrow y$ où $\Sigma_X^T \equiv (\Sigma_X \wedge \neg(T(\overline{y}) \wedge \neg T(y))) \vee (T(y) \wedge \neg T(\overline{y}))$ (Coste-Marquis et Marquis, 2023).

Toutefois, lorsque $T = R$ est une règle de classement, la définition ci-dessus de $\Sigma \star T$ peut être simplifiée. Ainsi :

Proposition 1 *Soit* $\Sigma = \Sigma_X \Leftrightarrow y$ *un circuit de classement sur* $X \cup \{y\}$*. Soit* $R = \varphi_X \Rightarrow y$ *(resp.* $R = \varphi_X \Rightarrow \overline{y}$*) une règle de classement sur* y *(resp. sur* $\overline{y}$*). On a* $\Sigma \star R \equiv (\Sigma_X \vee \varphi_X) \Leftrightarrow y$ *(resp.* $\Sigma \star R \equiv (\Sigma_X \wedge \neg\varphi_X) \Leftrightarrow y$*).*

La proposition suivante est un résultat clé de l'article. Elle montre que la rectification d'un circuit de classement par un ensemble de règles de classement déductibles d'un autre circuit de classement peut être réalisée *incrémentalement*, i.e., en rectifiant de façon itérative le premier circuit par chacune des règles considérées. Dans la suite, le circuit à rectifier sera de la forme $I \Leftrightarrow y$ où I est un arbre de décision sur X et le second circuit à partir duquel des règles de classement seront déduites sera de la forme $P \Leftrightarrow y$ où P un arbre optimisé sur X.

Proposition 2 *Soit* $\Sigma = \Sigma_X \Leftrightarrow y$ *un circuit de classement sur* $X \cup \{y\}$*. Soit* $\Phi = \Phi_X \Leftrightarrow y$ *un second circuit de classement sur* $X \cup \{y\}$*. Soit* $\{R_1, \ldots, R_k\}$ *un ensemble de règles de classement déductibles de* Φ*. On a* $\Sigma \star (R_1 \wedge \ldots \wedge R_k) \equiv (\Sigma \star R_1) \star \ldots \star R_k$*.*

Enfin, pour mettre en place notre approche, il faut montrer comment on peut extraire des règles de classement déductibles d'un circuit de classement $\Phi_X \Leftrightarrow y$, à partir d'explications abductives d'instances étant donné Φ_X. C'est le rôle de la proposition suivante :

Proposition 3 *Soit* $\Phi = \Phi_X \Leftrightarrow y$ *un circuit de classement sur* $X \cup \{y\}$*. Soit* $\boldsymbol{x} \in \boldsymbol{X}$ *une instance telle que* $\Phi_X(\boldsymbol{x}) = 1$ *(resp.* $\Phi_X(\boldsymbol{x}) = 0$*). Soit* t *une explication abductive pour* $\boldsymbol{x}$ *étant donné* Φ_X*.* $R = t \Rightarrow y$ *(resp.* $R = t \Rightarrow \overline{y}$*) est une règle de classement sur* y *(resp.* $\overline{y}$*) déductible de* Φ*.*

4 Distiller des arbres optimisés en arbres de décision

En combinant les propositions 1, 2 et 3, on peut définir un processus de distillation incrémental (et possiblement partiel) d'arbres optimisés P en arbres de décision I. L'idée est de déduire de $P \Leftrightarrow y$ des règles de classement de la forme $R = \varphi_X \Rightarrow y$ ou $R = \varphi_X \Rightarrow \overline{y}$ et de rectifier itérativement le circuit de classement $I \Leftrightarrow y$ considéré au départ par de telles règles. Lorsque φ_X est un arbre de décision sur X (ou quand $\varphi_X = t$ est un terme sur X), il est

possible de calculer $\Sigma \star R$ en temps polynomial comme un circuit de classement $I^R \Leftrightarrow y$ où I^R est un arbre de décision sur X (autrement dit, la rectification est alors une transformation XAI traitable pour les arbres de décision) (Coste-Marquis et Marquis, 2023).

Le caractère incrémental du processus est important : il ne s'agit pas de calculer d'abord l'ensemble de toutes les règles de classement déductibles de $P \Leftrightarrow y$ puis de rectifier $I \Leftrightarrow y$ par les règles obtenues. En effet, même si $P \Leftrightarrow y$ peut, comme tout autre classeur binaire, être caractérisé par une conjonction équivalente de règles de classement non conflictuelles $\bigwedge_{\boldsymbol{x}:P(\boldsymbol{x})=1}(t_{\boldsymbol{x}} \Rightarrow y) \wedge \bigwedge_{\boldsymbol{x}:P(\boldsymbol{x})=0}(t_{\boldsymbol{x}} \Rightarrow \overline{y})$, cette conjonction n'est pas directement utilisable car elle contient autant de règles que d'instances. Certaines règles pourraient être regroupées, mais, de manière générale, toute formule sous forme normale conjonctive (CNF) sur X équivalente à un arbre optimisé (ou, plus simplement, à une forêt aléatoire) est de taille exponentielle dans la taille de P dans le pire des cas. Par ailleurs, tout arbre de décision équivalent à une formule CNF sur X donnée est lui aussi de taille exponentielle dans la taille de cette formule CNF dans le pire des cas. Ces résultats d'efficacité spatiale des classeurs binaires ont été montrés dans (Audemard et al., 2022b; de Colnet et Marquis, 2023).

Pour cette raison, le mieux que nous puissions faire est de mettre en place une approche *paresseuse mais opportuniste* de distillation : le principe est de se focaliser sur les instances $\boldsymbol{x}$ qui sont mal classées par I (c'est-à-dire celles classées différemment par P) dès qu'elles apparaissent et de corriger I en dérivant pour chacune de ces instances $\boldsymbol{x}$ une règle de classement qui couvre $\boldsymbol{x}$ et est déductible de P.

Exemple 3 (suite de l'exemple 1) *Pour l'instance $\boldsymbol{x} = (0, 1, 1, 1)$, comme $I(\boldsymbol{x}) = 1$ et $P(\boldsymbol{x}) = 0$, nous devons corriger l'erreur de classement commise par I. En utilisant l'approche présentée dans (Audemard et al., 2023) nous calculons d'abord en temps polynomial l'explication abductive $t = \overline{x_1} = \overline{(S > 30)}$ pour $\boldsymbol{x}$ étant donné P. Comme $P(\boldsymbol{x}) = 0$, nous en dérivons directement le fait que $R = \overline{x_1} \Rightarrow \overline{y}$ soit une règle de classement déductible de $P \Leftrightarrow y$. On rectifie alors le circuit de classement $\Sigma = I \Leftrightarrow y$ par la règle R pour produire le circuit $\Sigma \star R$, qui équivaut à $(I \wedge \neg t) \Leftrightarrow y$ d'après la proposition 1. On construit donc un arbre de décision équivalent à $I^R = I \wedge \neg t$ en remplaçant chaque feuille 1 de I par un arbre équivalent à $\neg t$ (qui se construit en temps linéaire quand t est un terme), voir (Coste-Marquis et Marquis, 2023) pour plus de détails. On obtient l'arbre donné à la figure 2 (sous-figure de gauche), dans lequel le remplacement effectué aux feuilles 1 apparaît en bleu. Cet arbre est ensuite simplifié en utilisant la théorie du domaine (on obtient l'arbre de décision donné par la sous-figure de droite). Notons que la rectification réalisée a permis aussi de corriger l'erreur de classement qui était commise par $I \Leftrightarrow y$ sur l'instance $(0, 1, 0, 1)$. L'arbre de décision obtenu I^R classe toutes les instances de la même manière que P, sauf $(1, 1, 1, 0)$ qui sera éventuellement corrigée si l'instance $(1, 1, 1, 0)$ est considérée dans le futur.*

Il faut noter que si l'ordre dans lequel les rectifications par des règles de classement $R_1, \ldots, R_k$ ont lieu n'a pas d'impact sur le résultat obtenu (c'est ce qu'affirme la proposition 2), l'ordre dans lequel les instances $\boldsymbol{x}$ classées différemment par I et par P a, en général, une influence majeure sur les explications abductives qui vont être engendrées et donc aussi sur les règles de classement associées et les instances couvertes par ces règles. Si l'on ajoute à cette observation le fait que plusieurs explications abductives sont typiquement possibles pour chaque instance, le nombre d'instances qui restent mal classées après n étapes de rectifications peut varier grandement selon l'instance considérée et le choix d'explication réalisé à chaque étape.

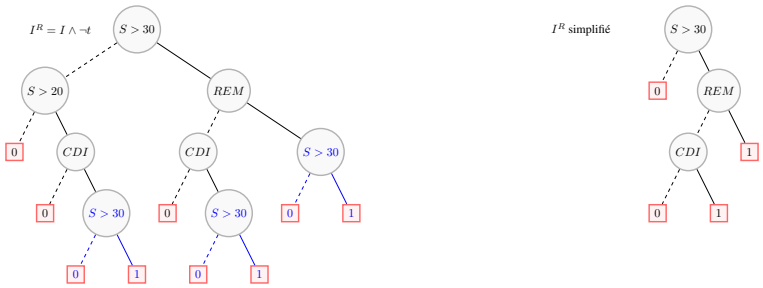


FIG. 2 – Un arbre de décision équivalent à $I \wedge \neg t$ (sous-figure de gauche). Un arbre de décision équivalent, obtenu par simplification (sous-figure de droite).

5 Expérimentations

Protocole expérimental Dans nos expérimentations, nous avons considéré divers jeux de données standards pour le classement binaire, issus de dépôts ouverts bien connus : UC Irvine Machine Learning Repository (UCI) [2] et openML [3]. Les instances de ces jeux de données sont représentées en utilisant des attributs de divers types (numériques, catégoriels ou booléens). Chaque instance est associée à une classe c qui vaut 1 ou 0 selon que l'instance est positive ou négative.

Chaque jeu de données a été partitionné en deux sous-ensembles, un ensemble utilisé seulement pour l'apprentissage des arbres (70% des instances) et un ensemble de validation (30% des instances) utilisé seulement pour mesurer leur précision empirique. Nous avons commencé par apprendre un modèle précis P, ici un arbre optimisé. Une fois P appris, nous avons réexprimé toutes les instances de chaque jeu de données dans l'espace des n conditions booléennes utilisées dans P (les instances obtenues sont ainsi décrites sur des attributs exclusivement booléens). A (resp. V) est l'ensemble des instances utilisées pour l'apprentissage (resp. l'ensemble des instances de validation) obtenues après binarisation. Nous avons appris 10 arbres de décision I à partir de A de façon à pouvoir réaliser une estimation empirique robuste par validation croisée à 10 blocs (en retenant 70% des instances de A pour l'entraînement de chaque arbre). La précision $\%C$ de chaque classeur utilisé C, mesurée empiriquement sur V, vaut $1 - \frac{|\{(\boldsymbol{x},c) \in V : C(\boldsymbol{x}) \neq c\}|}{|V|}$.

Les jeux de données utilisés sont décrits dans le tableau 1 : en plus du dépôt dont le jeu de données est issu, de la précision médiane $\%I$ obtenue pour les 10 arbres de décision considérés et de la précision $\%P$ de l'arbre optimisé, la tableau indique les valeurs de $|E|$, le nombre d'instances du jeu de données, $|F|$, le nombre d'attributs servant à décrire initialement les instances, $|B|$, le nombre de conditions booléennes utilisées dans le jeu de données une fois binarisé en utilisant P et enfin $|V_I^{\pm}|$ le nombre d'instances de l'ensemble de validation V classées différemment par I et P.

La phase suivante a été de corriger les arbres de décision I chaque fois que nécessaire, en utilisant P comme oracle pour la vérité de terrain. Nous avons considéré, pour chaque jeu de données, les instances de V en éliminant systématiquement parmi elles les instances impos-

2. `https://archive.ics.uci.edu/ml/index.php`
3. `https://www.openml.org/`

Jeu de données	$\|E\|$	$\|F\|$	$\|B\|$	$\%I$	$\%P$	Dépôt	$\|V_I^{\pm}\|$
bank	4521	48	742	87.9%	89.97%	UCI	71
compas	6172	11	43	68.63%	69.29%	openML	128
bupa	345	5	103	83.33%	97.26%	UCI	8
contraceptive	1473	21	77	64.07%	67.74%	UCI	124
balance-scale	625	4	17	89.69%	90.42%	UCI	7
australian	690	38	364	81.63%	86.20%	openML	24
biodegradation	1054	41	727	82.53%	88.64%	openML	41
german	1000	58	131	92.85%	96.33%	UCI	18
cleveland	303	23	145	76.73%	87.5%	openML	15

TAB. 1 – Description des jeux de données et précisions empiriques avant toute correction.

sibles (on utilise pour cela la théorie du domaine dérivée des attributs de départ). Pour chaque $(\boldsymbol{x}, c)$ de V, $P(\boldsymbol{x})$ est considéré comme la « vraie » classe de $\boldsymbol{x}$, donc $V_I^{\pm} = \{(\boldsymbol{x}, c) \in V : I(\boldsymbol{x}) \neq P(\boldsymbol{x})\}$ est le sous-ensemble des instances de V qui, selon P, doivent être corrigées. La précision I_P de I *relativement à* P, mesurée empiriquement sur V, vaut $1 - \frac{|V_I^{\pm}|}{|V|}$. de sorte que si $I(\boldsymbol{x}) = P(\boldsymbol{x})$ pour chaque $(\boldsymbol{x}, c) \in V$, I_P vaut 100%. Ainsi, $1 - I_P$ indique la proportion du nombre d'instances de V qui restent à corriger.

Pour chaque $\boldsymbol{x} \in V_I^{\pm}$, nous avons calculé une explication abductive t pour $\boldsymbol{x}$ étant donné P en utilisant la bibliothèque PyXAI (`https://www.cril.univ-artois.fr/pyxai/`), puis

— dans l'approche par ré-entraînement, nous avons construit un échantillon d'instances $\boldsymbol{x}'$ couvertes par t et contenant $\boldsymbol{x}$. Outre $\boldsymbol{x}$, nous avons fixé un plafond (égal à 16) pour la taille de l'échantillon obtenu, de façon à ne pas faire croître de façon trop importante l'ensemble d'apprentissage utilisé à chaque étape. Cet échantillon est obtenu en choisissant au hasard selon une distribution uniforme quatre conditions de P ne figurant pas dans t et parmi les instances générées, seules celles qui sont réalisables (Gorji et Rubin, 2022) sont conservées. Pour chaque $\boldsymbol{x}'$ de l'échantillon, $(\boldsymbol{x}', P(\boldsymbol{x}))$ a été ajouté à l'ensemble d'apprentissage considéré initialement et nous avons procédé à un nouvel apprentissage du modèle I en utilisant cet ensemble étendu.

— dans l'approche par rectification, nous avons rectifié l'arbre de décision courant par la règle de classement $R = t \Rightarrow y$ quand $P(\boldsymbol{x}) = 1$ et par la règle de classement $R = t \Rightarrow \overline{y}$ quand $P(\boldsymbol{x}) = 0$, ce qui a conduit à un nouvel arbre de décision.

Après chaque correction, nous avons mis $V_I^{\pm}$ à jour avant de passer à la correction suivante (cela est nécessaire puisque I a été modifié).

Pour chacune des deux approches de correction, nous avons mesuré après chaque correction (donc pour chaque arbre de décision I obtenu) :

— la précision I_P de l'arbre de décision I relativement à P (estimée sur V),

— la taille de l'arbre de décision I (en nombre de nœuds N) et sa profondeur H.

Les arbres de décision I ont été appris en utilisant l'algorithme fourni dans la bibliothèque Scikit-Learn (Pedregosa et al., 2011), en ajustant les valeurs des hyperparamètres pour obtenir une bonne précision empirique et éviter le sur-apprentissage. Les arbres optimisés P ont été appris en utilisant l'algorithme fourni dans la bibliothèque XGBoost (Chen et Guestrin, 2016), en ajustant les valeurs des hyperparamètres pour tenter d'obtenir une précision empirique meilleure que celles obtenues par les arbres de décision sur le même jeu de données.

Jeu de données		Étapes 0 I_P	N	H	1 I_P	N	H	2 I_P	N	H		étape finale (rec) I_P	N	H	f
bank	rec	94.76	27	5	94.87	58	20	94.99	86	21		100.0	3447	34	45
	ren	94.76	27	5	94.47	27	5	94.29	27	5		93.71	29	5	-
compas	rec	93.08	63	7	93.16	70	9	93.87	75	10		100.0	227	15	39
	ren	93.08	63	7	93.08	63	7	93.54	63	7	...	93.25	65	7	-
bupa	rec	92.30	42	8	93.26	60	13	94.23	76	14		100.0	157	14	8
	ren	92.30	42	8	91.34	47	8	91.82	51	8		91.34	53	8	-
contraceptive	rec	71.83	53	7	72.05	68	14	72.28	90	14		100.0	3027	26	123
	ren	71.83	53	7	72.39	56	7	72.05	54	7		77.14	59	7	-
balance_scale	rec	96.01	32	7	96.54	39	9	97.07	51	10	...	100.0	83	12	7
	ren	96.01	32	7	95.74	34	7	96.27	33	7		96.80	35	7	-
australian	rec	88.40	37	6	88.88	56	14	89.37	99	14		100.0	1357	22	24
	ren	88.40	37	6	88.88	36	6	90.09	38	6		92.27	31	6	-
biodegradation	rec	86.79	13	4	87.73	67	31	88.67	257	35	...	100.0	12524	51	15
	ren	86.79	13	4	87.26	13	4	88.67	13	4		86.79	13	4	-
german	rec	94.0	72	9	94.33	117	18	94.66	161	18		100.0	1280	24	18
	ren	94.0	72	9	94.33	73	9	94.16	71	9		95.5	84	9	-
cleveland	rec	82.96	68	8	85.71	98	11	86.81	110	12		100.0	284	14	13
	ren	82.96	68	8	84.61	65	8	86.81	71	8		88.46	70	8	-

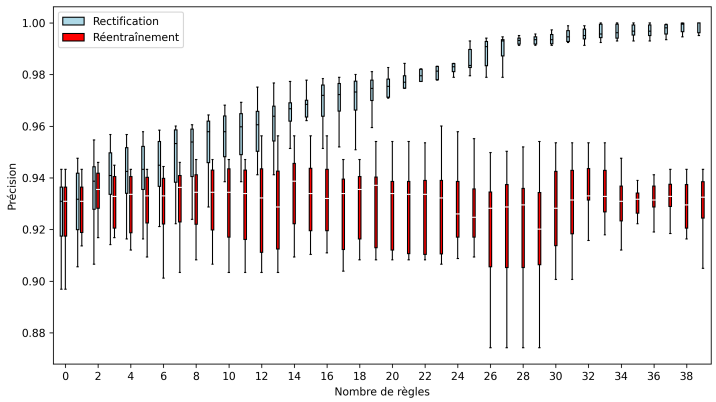
TAB. 2 – Précision empirique I_P relativement à P (en %), nombre médian de nœuds N et profondeur médiane H des arbres de décision pour différents jeux de données et pour chaque méthode de correction. f est le nombre d'étapes au bout duquel $V_I^{\pm}$ devient vide lorsque la correction est réalisée par rectification.

Résultats empiriques Le tableau 2 donne, pour chaque jeu de données, les valeurs médianes des précisions empiriques I_P (en %) relativement à P et des nombres de nœuds N de l'arbre corrigé pour chacune des deux méthodes de correction (rec est la rectification, ren le ré-entraînement). On peut observer qu'avant toute correction (i.e., à l'étape 0) les valeurs de I_P diffèrent de celles de $\%I$ données dans le tableau 1.

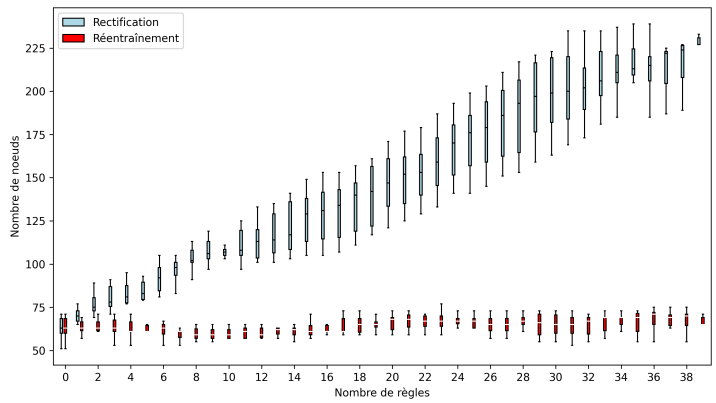
Trois étapes de correction sont mises en évidence. Après chaque correction par rectification, *il est garanti que I_P augmente strictement* puisqu'on est sûrs qu'au moins l'instance qui a déclenché cette étape de correction a été corrigée. Le nombre d'étapes nécessaires f pour corriger *totalement* l'arbre de décision en le rectifiant est donc bien défini. Quand on met en balance ce nombre f avec le nombre $|V_I^{\pm}|$ d'instances initiales mal classées comme donné dans le tableau 1, on observe l'impact que la rectification avec des règles couvrant un nombre possiblement élevé d'instances peut avoir (par exemple, pour `compas`, 39 étapes suffisent alors que 128 instances sont mal classées au départ).

A contrario, les résultats empiriques obtenus montrent que les corrections successives par ré-entraînement peuvent conduire à faire décroître la précision de l'arbre de décision relativement à l'arbre optimisé, rendant une *correction complète impossible*. En effet, l'ajout de $(\boldsymbol{x}, P(\boldsymbol{x}))$ à l'ensemble d'apprentissage avant ré-entraînement de I ne garantit pas, en général, que $\boldsymbol{x}$ sera classé comme demandé par P après le ré-entraînement. En outre, une instance mal classée et corrigée à une certaine étape par ré-entraînement peut se retrouver à nouveau mal classée suite à un ré-entraînement ultérieur.

Le tableau 2 montre aussi que la croissance de la taille des arbres (et celle de leur profondeur) est plus importante quand la rectification est utilisée, mais elle reste très souvent raisonnable : on arrive ainsi à construire en pratique les arbres rectifiés en un temps très court (nous n'avons pas fait figurer dans le tableau 2 les temps de calcul requis pour réaliser la correction car ceux-ci restent réduits – au plus 2s et en général moins de 0.1s – quelle que soit l'approche de correction utilisée).



(a) Précision empirique I_P obtenue après rectification ou ré-entraînement.



(b) Nombre de nœuds de l'arbre de décision obtenu après rectification ou ré-entraînement.

FIG. 3 – Comparaison de la correction d'un arbre de décision par rectification et par ré-entraînement sur le jeu de données `compas`.

La figure 3 fournit des statistiques plus détaillées concernant les résultats empiriques obtenus pour le jeu de données `compas`. Les distributions des précisions et des tailles d'arbre sont résumées sous forme de boîtes à moustaches. Le nombre d'étapes de correction présenté (39) est beaucoup plus grand que celui mis en avant dans le tableau 2 (limité aux deux premières étapes et à la dernière étape pour l'approche par rectification de façon à tenir dans la largeur

de la page). Les résultats obtenus sur `compas` sont en phase avec ceux donnés dans le tableau 2 : la figure met bien en évidence l'accroissement de la précision relative permis par la rectification (en comparaison avec le ré-entraînement) au prix d'un accroissement plus important de la taille de l'arbre de décision, qui reste gérable ici (la médiane de la taille des arbres obtenus au bout de 39 rectifications est grosso modo triple de celle de départ).

6 Conclusion

Nous avons proposé une approche de distillation d'arbres optimisés P en arbres de décision I basée sur l'opération de rectification. Une telle approche vise à aboutir à un arbre de décision offrant un compromis acceptable en terme de précision et d'interprétabilité. Nous avons montré empiriquement que l'approche proposée fournit des résultats intéressants par comparaison à la distillation par ré-entraînement. Par rapport à une approche de distillation où il s'agirait de traduire P en I « en bloc », notre approche est paresseuse et opportuniste : on ne corrige I que lorsque cela se révèle nécessaire, c'est-à-dire quand on rencontre une instance $\boldsymbol{x}$ telle que $I(\boldsymbol{x}) \neq P(\boldsymbol{x})$. Cette façon de procéder permet de contrôler la taille de l'arbre de décision I résultant (rectifié), donc d'arrêter le processus de rectification quand on le souhaite.

Une perspective de ce travail consiste à développer un algorithme de minimisation d'arbre de décision, qui pourra être utilisé dans le processus de distillation par rectification lorsque la taille de l'arbre obtenu devient trop importante. Comme cette tâche de minimisation est NP-difficile (Sieling, 2008), il faudra évaluer l'algorithme pour déterminer dans quelle mesure il est suffisamment efficace en pratique.

Remerciements Merci aux relecteurs anonymes pour leurs remarques qui ont été très utiles pour améliorer l'article. Le travail correspondant a été réalisé dans le cadre de la chaire ANR d'enseignement et de recherche EXPEKCTATION (ANR-19-CHIA-0005-01).

Références

Asadulaev, A., I. Kuznetsov, et A. Filchenkov (2019). Interpretable few-shot learning via linear distillation. *CoRR abs/1906.05431*.

Audemard, G., S. Bellart, L. Bounia, F. Koriche, J. Lagniez, et P. Marquis (2021). On the computational intelligibility of boolean classifiers. In *Proc. of KR'21*, pp. 74–86.

Audemard, G., S. Bellart, L. Bounia, F. Koriche, J. Lagniez, et P. Marquis (2022a). On the explanatory power of boolean decision trees. *Data Knowl. Eng. 142*, 102088.

Audemard, G., S. Bellart, L. Bounia, F. Koriche, J. Lagniez, et P. Marquis (2022b). Trading complexity for sparsity in random forest explanations. In *Proc. of AAAI'22*, pp. 5461–5469.

Audemard, G., J. Lagniez, P. Marquis, et N. Szczepanski (2023). Computing abductive explanations for boosted trees. In *Proc. of AISTATS'23*, pp. 4699–4711.

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

Chen, T. et C. Guestrin (2016). XGBoost : A scalable tree boosting system. In *Proc. of KDD'16*, pp. 785–794.

Coste-Marquis, S. et P. Marquis (2021). On belief change for multi-label classifier encodings. In *Proc. of IJCAI'21*, pp. 1829–1836.

Coste-Marquis, S. et P. Marquis (2023). Rectifying binary classifiers. In *Proc. of ECAI'23*, pp. 485–492.

de Colnet, A. et P. Marquis (2023). On translations between ML models for XAI purposes. In *Proc. of IJCAI'23*, pp. 3158–3166.

Freund, Y. et R. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci. 55*(1), 119–139.

Frosst, N. et G. E. Hinton (2017). Distilling a neural network into a soft decision tree. *CoRR abs/1711.09784*.

Gorji, N. et S. Rubin (2022). Sufficient reasons for classifier decisions in the presence of domain constraints. In *Proc. of AAAI'22*, pp. 5660–5667.

Gou, J., B. Yu, S. J. Maybank, et D. Tao (2021). Knowledge distillation : A survey. *Int. J. Comput. Vis. 129*(6), 1789–1819.

Hinton, G., O. Vinyals, et J. Dean (2015). Distilling the knowledge in a neural network. *CoRR abs/1503.02531*.

Ignatiev, A., N. Narodytska, et J. Marques-Silva (2019). Abduction-based explanations for machine learning models. In *Proc. of AAAI'19*, pp. 1511–1519.

Molnar, C. (2019). *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*. Leanpub.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, et E. Duchesnay (2011). Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

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

Sieling, D. (2008). Minimization of decision trees is hard to approximate. *J. Comput. Syst. Sci. 74*(3), 394–403.

Wood-Doughty, Z., I. Cachola, et M. Dredze (2022). Model distillation for faithful explanations of medical code predictions. In *Proc. of BioNLP@ACL'22*, pp. 412–425.

Zhou, Z.-H. (2019). Abductive learning : towards bridging machine learning and logical reasoning. *Science China Information Science 62*(7), 76101 :1–76101 :3.

Summary

In this paper we present an approach to the distillation of boosted trees into decision trees. Such a distillation process aims to derive an ML model offering a acceptable trade-off in terms of precision and interpretability. We explain how the approach to correction of binary classifiers, called rectification, and introduced recently can be used to implement such a distillation process. We show empirically that this approach provides interesting results, compared to distillation by re-training.