ECAI 2024
U. Endriss et al. (Eds.)
© 2024 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA240601

Locally-Minimal Probabilistic Explanations

Yacine Izza[a,*], **Kuldeep S. Meel**[b] **and Joao Marques-Silva**[c]

[a]CREATE, NUS, Singapore
[b]University of Toronto, Canada
[c]ICREA, University of Lleida, Spain

Abstract. Explainable Artificial Intelligence (XAI) is widely regarded as a cornerstone of trustworthy AI. Unfortunately, most work on XAI offers no guarantees of rigor. In high-stakes domains, e.g. uses of AI that impact humans, the lack of rigor of explanations can have disastrous consequences. Formal abductive explanations offer crucial guarantees of rigor and so are of interest in high-stakes uses of machine learning (ML). One drawback of abductive explanations is explanation size, justified by the cognitive limits of human decision-makers. Probabilistic abductive explanations (PAXps) address this limitation, but their theoretical and practical complexity makes their exact computation most often unrealistic. This paper proposes novel efficient algorithms for the computation of locally-minimal PAXps, which offer high-quality approximations of PXAps in practice. The experimental results demonstrate the practical efficiency of the proposed solutions.

1 Introduction

For more than a decade, progress in Machine Learning (ML) has been revolutionary. However, such progress is most often achieved through the use of highly complex ML models, which human decision-makers are unable to comprehend. In recent years, eXplainable Artificial Intelligence (XAI) has been proposed as a solution to deliver trustworthy AI, thus enabling the use of complex ML models in high-stakes domains. Unfortunately, most work on XAI offer little to no guarantees of rigor. Unsurprisingly, there exists growing evidence to the limitations and perils of deploying XAI solutions that offer no guarantees of rigor [18, 15, 34, 36, 13].

Recent work has advocated the use of formal, logic-based explanations [52, 17, 37, 9]. Formal XAI offers the strongest guarantees of rigor, but it also exhibits important limitations. One of the best-known limitations is scalability, due to the complexity of logic-based reasoning. While some progress has been observed for tree ensembles [21, 19, 28] among others, neural networks have represented a significant challenge for formal XAI [17]. Nevertheless, recent work [62, 27] showed important progress towards achieving scalability of formal XAI in deep Neural Network models. Another key limitation of formal XAI is the size of explanations, which can often be well beyond the cognitive limits of human decision makers [43]. Probabilistic explanations [59] aim at finding explanations of smaller size, and so represent one proposed solution for the size limitation of formal explanations. Probabilistic abductive explanations trade off the strong theoretical guarantees of rigor of abductive explanations for smaller explanation sizes, while still ensuring the quality of approximate explanations. Unfortunately, the complexity of finding probabilistic abductive explanations is in general unwiedly [59], e.g. a practical algorithm should be expected to call a counting oracle exponentially many times. Even for restricted families of classifiers, e.g. decision trees, it is known that the computation of probabilistic abductive explanations is computationally hard, albeit solvable with a limited number of calls to an NP-oracle [24, 1, 25]. Clearly, the case of decision trees, while still computationally hard, is drastically easier to solve in practice.

One key difficulty in computing probabilistic abductive explanations results from the non-monotonicity of the sets representing or not a probabilistic explanation. This lack of monotonicity requires the analysis of all possible subsets of features in order to find a subset-minimal set. In such a situation, one approximation to subset-minimality is local minimality, i.e. sets that are minimal if at most one feature is allowed to be removed. A surprising experimental observation [25] is that, in the case of decision trees and other graph-based classifiers, locally-minimal explanations are in most cases also subset-minimal explanations — reported results show for decision trees that in 99.8% cases computed approximate explanations are subset minimal.

Given these earlier empirical observations, one solution towards devising practical algorithms for approximating probabilistic abductive explanations is to compute locally-minimal explanations, thereby eliminating the need to analyze all possible subsets of a target set of features. This paper proposes two new techniques for computing approximate locally-minimal explanations, one based on approximate model counting, and the other based on sampling with probabilistic guarantees. The experimental results support the practical efficiency of the proposed methods for two case studies of classifiers random forest (RF) and binarized neural network (BNN) — results on BNNs show that explanation length of our solution drops by one third to two third of abductive explanation size. (An earlier version of this work exists as a CoRR report in [26].)

The paper is organized as follows. Section 2 introduces the notation and definitions used in the paper. Section 3 overviews the logic encodings of classifiers used in later sections. Section 4 describes the approach proposed in this paper. The experimental results are discussed in Section 5. Finally, the paper concludes in Section 6.

2 Background

Here we introduce the notation and background on formal XAI and the case study of two classifier families, which enable (Boolean) logical encoding.

* Corresponding Author. Email: izza@comp.nus.edu.sg.

2.1 Classification problems

This paper considers classification problems, which are defined on a set of features (or attributes) $\mathcal{F} = \{1, \ldots, m\}$ and a set of classes $\mathcal{K} = \{c_1, c_2, \ldots, c_K\}$. Each feature $i \in \mathcal{F}$ takes values from a domain $\mathbb{D}_i$. In general, domains can be categorical or ordinal, with values that can be boolean, integer or real-valued. Feature space is defined as $\mathbb{F} = \mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_m$; $|\mathbb{F}|$ represents the total number of points in $\mathbb{F}$. For boolean domains, $\mathbb{D}_i = \{0, 1\} = \mathbb{B}$, $i = 1, \ldots, m$, and $\mathbb{F} = \mathbb{B}^m$. The notation $\mathbf{x} = (x_1, \ldots, x_m)$ denotes an arbitrary point in feature space, where each x_i is a variable taking values from $\mathbb{D}_i$. The set of variables associated with features is $X = \{x_1, \ldots, x_m\}$. Moreover, the notation $\mathbf{v} = (v_1, \ldots, v_m)$ represents a specific point in feature space, where each v_i is a constant representing one concrete value from $\mathbb{D}_i$.

An ML classifier $\mathbb{M}$ is characterized by a (non-constant) *classification function* κ that maps feature space $\mathbb{F}$ into the set of classes $\mathcal{K}$, i.e. $\kappa : \mathbb{F} \to \mathcal{K}$. An *instance* denotes a pair $(\mathbf{v}, c)$, where $\mathbf{v} \in \mathbb{F}$ and $c \in \mathcal{K}$, with $c = \kappa(\mathbf{v})$. Given the above, we represent a classifier $\mathcal{M}$ by a tuple $\mathcal{M} = (\mathcal{F}, \mathbb{F}, \mathcal{K}, \kappa)$.

Moreover, and given a concrete instance $(\mathbf{v}, c)$, an explanation problem is represented by a tuple $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$.

2.2 Binarized Neural Network (BNNs)

Binarized Neural Networks (BNNs) [14, 8] are a widely used family of neural networks. BNNs exhibit a number of important features that allow them to be deployed into embedded devices [41, 31]. Concretely, a BNN is composed of a number of layers of neurons. The neurons of the intermediate layers (hidden layers) compute a mapping function: $\{-1, 1\}^n \to \{-1, 1\}^m$ on input $x^i \in \{-1, 1\}^n$, whilst the neurons of the last layer (output layer) map a binary tensor to real domain: $\{-1, 1\}^{n_d} \to \mathbb{R}^K$ on input $x^i \in \{-1, 1\}^{n_d}$. Furthermore, outputs of intermediate layers are obtained with applying three transformations on the input tensor x: a linear transformation (LIN), batch normalization (BatchNorm) and binarization (BIN), such that $x^{i+1} = \textsc{Bin}(\textsc{BatchNorm}(\textsc{Lin}(x)))$, where: $\textsc{Lin}(x^i) = W^i x^i + b^i$, where $W^i \in \{-1, 1\}^{m \times n}$ and $b^i \in \mathbb{R}^n$, $\textsc{BatchNorm}(y_j) = \alpha^i_j \left(\frac{y_j - \mu^i_j}{\sigma^i_j} \right)$, where $\alpha^i_j, \mu^i_j, \sigma^i_j, y_j \in \mathbb{R}^m$, and $\textsc{Bin}(z) = sign(z) \in \{-1, 1\}^m$ and $z \in \mathbb{R}^m$. Lastly, the output layer applies a linear transformation before an *argmax* mapping that picks a class to predict. (Note that [46, 29] propose to use a ternary neural network in order to generate sparse matrices, thus $W^i \in \{-1, 0, 1\}$.)

2.3 Random Forests (RFs)

Random Forests (RFs) [4, 65, 68, 12, 11, 69] are an example of tree ensemble ML models, which find a wide range of practical applications. Although RFs are known to lack in interpretability, recent work [21, 20] demonstrates that large tree ensembles can be efficiently analyzed with logical-based reasoners like SAT/SMT oracles, and so can be explained. Conceptually, an RF is collection of n decision trees (DTs), where each tree $\mathcal{T}_i$ of the ensemble $\mathcal{T}$ is trained on a randomly selected subset of the training data so that the trees of the RF are not correlated. (In contrast to a single DT, RFs are less prone to over-fitting and so offer in general better accuracy on test data [4].) Similar to the original proposal for RFs [4], the predictions of an RF classifier are made by majority vote of the trees, that is each tree predicts a class and the class with largest score is picked.[1] Other prediction function could be considered, e.g. weighted voting implemented in different learning tools like Scikit-learn [48].

[1] In case of ties of votes, the lexicographically smallest class is the commonly picked one.

2.4 Logic-based Explainability

Formal explainability aims to answer the question "*why*" a prediction is made by a classifier by identifying a selection of the input features that are responsible for the prediction, and consequently can be referred to as *feature selection* explanations. More recently, formal *feature attribution* explanations [66, 2, 67, 33] have been introduced, which provides the importance score of each feature included in the explanation set. We overview the formal definition of both type of explanations.

Abductive explanations. Abductive (PI-) explanations [52, 17] denote a minimal set of literals (relating a feature value x_i and a constant $v_i \in \mathbb{D}_i$) that are sufficient for the prediction. Formally, given an explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$, a set of features $\mathcal{X} \subseteq \mathcal{F}$ is a *weak abductive explanation* [7] (weak AXp) if the following predicate holds true:

$$\mathsf{WeakAXp}(\mathcal{X}) := \forall(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i) \right] \rightarrow (\kappa(\mathbf{x}) = c) \quad (1)$$

The classifier assigns the same class to all feature-vectors which agree with $\mathbf{v}$ on features in a weak AXp $\mathcal{X}$. Such sets $\mathcal{X}$ may contain features which do not contribute to the decision, which leads to the following definition. A set of features $\mathcal{X} \subseteq \mathcal{F}$ is an *abductive explanation* (AXp) if the following predicate holds true:

$$\mathsf{AXp}(\mathcal{X}) := \mathsf{WeakAXp}(\mathcal{X}) \land \forall(\mathcal{X}' \subsetneq \mathcal{X}). \neg \mathsf{WeakAXp}(\mathcal{X}') \quad (2)$$

Clearly, an AXp is any weak AXp that is subset-minimal (or irreducible). It is straightforward to observe that the definition of predicate $\mathsf{WeakAXp}$ is monotone, and so an AXp can instead be defined as follows:

$$\mathsf{AXp}(\mathcal{X}) := \mathsf{WeakAXp}(\mathcal{X}) \land \forall(j \in \mathcal{X}). \neg \mathsf{WeakAXp}(\mathcal{X} \setminus \{j\}) \quad (3)$$

Observe that an instance $(\mathbf{v}, c)$ can admit more that one AXp (which are worst-case exponential in number). We denote by $\mathbb{A}(\mathbf{v}, c)$ the set of all possible AXp's of $(\mathbf{v}, c)$.

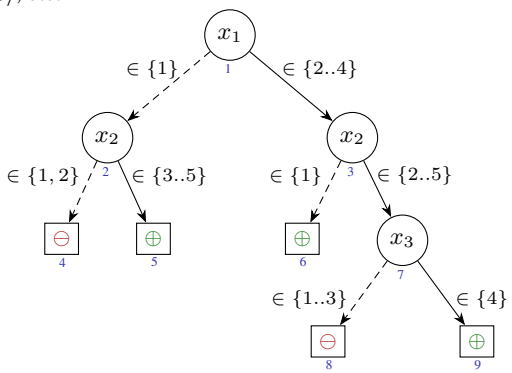
Logic-based explainability is covered in a number of recent works (see surveys [37, 9, 38, 35]). Besides, probabilistic explanations are investigated in [59, 23, 24, 1, 25, 3, 22].

Running Example. We consider a decision tree (DT) as a running example to illustrate the different class of explanations considered in this work (namely, AXp's and Probabilistic AXp's).

Example 1. Figure 1 shows an example decision tree, adapted from a toy example shown in [25], that admits a vector input of three features $\{x_1, x_2, x_3\}$ where the feature domains are, respectively, $\mathbb{D}_1 = \mathbb{D}_3 = \{1, \ldots, 4\}$ and $\mathbb{D}_2 = \{1, \ldots, 5\}$, and predicts class $\oplus$ or $\ominus$. Consider instance $\mathbf{v} = (2, 3, 1)$, with $c = \kappa(\mathbf{v}) = \ominus$. (The point $\mathbf{v}$ is consistent with the path of nodes $\langle 1, 3, 7, 8 \rangle$). Clearly, one can observe that $\mathcal{X} = \{1, 2, 3\}$ is an AXp for $\mathbf{v}$. i.e. observe that if we drop x_1, then $(x_2 = 3 \land x_3 = 1)$ is consistent with the path $\langle 1, 2, 5 \rangle$, or x_2 then $(x_1 = 2 \land x_3 = 1)$ is consistent with $\langle 1, 3, 6 \rangle$, or x_3 then $(x_1 = 2 \land x_2 = 3)$ is consistent with $\langle 1, 3, 7, 9 \rangle$.

Formal Feature Attribution AXp's. Formal feature attribution (FFA) [66, 2] abductive explanations are an extension of (plain) abductive explanations, in the sense that they provide additional information about the importance score of the features in the AXp's. Feature attribution AXp can be seen as an aggregation of all possible (or

Figure 1: DT example, s.t. nodes of the tree are enumerated from 1 to 9; a tree path is represented by a sequence of nodes, e.g. $\langle 1, 2, 4\rangle$, $\langle 1, 3, 6\rangle$, etc.



partial collection in the context of approximate FFA) AXp's [2] for a given instance $\mathbf{v}$ to be explanations. Concretely, FFA of a feature $\forall i \in \mathcal{F}$, denoted $\mathsf{ffa}(i)$ is defined as the proportion of AXp's where it occurs, i.e.

$$\mathsf{ffa}(i) = \frac{|\{\mathcal{X} \mid \mathcal{X} \in \mathbb{A}(\mathbf{v}, c), i \in \mathcal{X}\}|}{\mathbb{A}(\mathbf{v}, c)}$$

We can define now a *formal feature attribution* abductive explanation (FFAXp) as a set of features $\mathcal{X} \subseteq \mathcal{F}$ for which $i \in \mathcal{X}, \mathsf{ffa}(i) > 0$ and $i \in \mathcal{F} \setminus X, \mathsf{ffa}(i) = 0$. Clearly, an FFAXp $\mathcal{X}$ is also a weak AXp but not necessarily an AXp. Note that in practice FFAXp's are (much) larger than AXp's, accordingly our interest is to seek for approximately succinct FFAXp's.

2.5 Formal Probabilistic Explanations

We follow the recent accounts on formal probabilistic explanations [25].

Probabilistic AXp. A *weak probabilistic* AXp (weak PAXp) [24, 25] for a given explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$, is a a set of features fixed to values of $\mathbf{v}$ for which the conditional probability of predicting the correct class c is no less than a given threshold $\mathrm{T} \in [0, 1]$. Thus, $\mathcal{X} \subseteq \mathcal{F}$ is a weak PAXp if the following predicate holds true,

$$\begin{aligned}\mathsf{WeakPAXp}(\mathcal{X}) &:= \Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = c \,|\, \mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}}) \geq \mathrm{T} \qquad (4)\\ &:= \frac{|\{\mathbf{x} \in \mathbb{F} : \kappa(\mathbf{x}) = c \land (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}|}{|\{\mathbf{x} \in \mathbb{F} : (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}|} \geq \mathrm{T}\end{aligned}$$

which means that the fraction of the number of points predicting the target class and consistent with the fixed features (represented by $\mathcal{X}$), given the total number of points in feature space consistent with the fixed features, must exceed T. (Note that $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = c \,|\, \mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})$ is commonly referred as the probability, precision or accuracy of $\mathcal{X}$ [51, 60, 59], thus we will use these terms interchangeably throughout the paper. For notation simplicity, we will also denote by $\Pr_{\mathbf{x}}(\mathcal{X})$ the probability/precision of an explanation $\mathcal{X}$ for $\mathcal{E}$.) Moreover, a set $\mathcal{X} \subseteq \mathcal{F}$ is a *probabilistic* AXp (or (plain) PAXp) if the following predicate holds true,

$$\begin{aligned}\mathsf{PAXp}(\mathcal{X}) &:= \mathsf{WeakPAXp}(\mathcal{X}) \land \qquad (5)\\ &\quad \forall(\mathcal{X}' \subsetneq \mathcal{X}).\neg\mathsf{WeakPAXp}(\mathcal{X}')\end{aligned}$$

Thus, $\mathcal{X} \subseteq \mathcal{F}$ is a PAXp if it is a weak PAXp that is also subset-minimal.

As demonstrated in [24], weak PAXp (see Eq. (4)) does not guarantee monotonicity. In turn, this makes the computation of (subset-minimal) PAXp's harder. With the purpose of identifiying classes of weak PAXp's that are easier to compute, it will be convenient to study *locally-minimal* PAXp's.

Locally-minimal Probabilistic AXp. A set of features $\mathcal{X} \subseteq \mathcal{F}$ is a locally-minimal PAXp if,

$$\begin{aligned}\mathsf{LmPAXp}(\mathcal{X}) &:= \mathsf{WeakPAXp}(\mathcal{X}) \land \qquad (6)\\ &\quad \forall(j \in \mathcal{X}).\neg\mathsf{WeakPAXp}(\mathcal{X} \setminus \{j\})\end{aligned}$$

As observed earlier, because the predicate $\mathsf{WeakAXp}$ is monotone, subset-minimal AXp's match locally-minimal AXp's. An important practical consequence is that most algorithms for computing one subset-minimal AXp will instead compute a locally-minimal AXp, since these will be the same. Nevertheless, a critical observation is that in the case of probabilistic AXp's (see Eq. (4)), the predicate $\mathsf{WeakPAXp}$ is *not* monotone. Thus, there can exist locally-minimal PAXp's that are not subset-minimal PAXp's.

An easy observation is that LmPAXp definition also applies for FFAXp's. We denote by LmPFFAXp's the *locally-minimal probabilistic* FFAXp's to distinguish between probabilistic AXp's and FFAXp's.

Finally, the predicate $\mathsf{MinPAXp}$ defines the *minimum-size* PAXp's (or a smallest PAXp's). Note that in practice, explanation size of PAXp's and LmPAXp's are often not much larger than minimum-size PAXp's. This observation is supported by the empirical results on different families of classifiers shown in [25].

Example 2. Consider again DT and instance $\mathbf{v}$ of Example 1. Table 1 summarizes the computation of $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \ominus | (\mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}}))$ for different sets of features $\{1, 2, 3\}$, $\{1, 3\}$, $\{2\}$ and $\{3\}$. Moreover, it reports whether the analyzed subset $\mathcal{S}$ is a (weak) AXp, (Weak) PAXp, a MinPAXp or an LmPAXp. (The analysis of all points in feature space is omitted for brevity.) The set $\{1, 2\}$ is an LmPAXp and a PAXp (and necessarily a weak PAXp) since the he probability of predicting $\ominus$ is greater than T and its subsets $\{1\}$ and $\{2\}$ are not weak PAXp. The set $\{1, 3\}$ and $\{2, 3\}$ are weak PAXp but not PAXp or LmPAXp, since that their respective probabilities are are greater than T but the subset $\{3\}$ is also a weak PAXp. Moreover, $\{3\}$ is a minimum PAXp (and necessarily a PAXp). Observe that the order in which the features are analyzed when extracting an LmPAXp is important in order to find the smallest explanation. Assume for instance we traverse the features in the order $\langle 3, 2, 1\rangle$. Then, we can easily see that the final LmPAXp would be $\mathcal{X} = \{1, 2\}$ of size 2 and not $\mathcal{X} = \{3\}$ of size 1.

3 Logic Encodings

In this section we detail the encodings for both binarized neural networks and random forests (with majority voting). It should be noted that the proposed declarative encodings on both RFs and BNNs are polynomial on the size of the original ML classifier.

BNN Encoding. Recent work [45, 6, 46, 29] on formal verification of BNNs have proposed to formulate a number of verification queries on BNNs into propositional formulas or pseudo-Boolean (PB) formulas. We extend the SAT/PB -based encoding for robustness checking for computing explanation in BNNs. We underline that PB constraints can be efficiently converted into CNF formulas. Therefore, this work focuses on PB encodings, which can be translated into SAT, instead of an immediate SAT encoding. Roughly, encoding a BNN is

Table 1: Examples of sets of fixed features given instance $\mathbf{v} = (2, 3, 1)$ (s.t. $\kappa(\mathbf{v}) = \ominus$) and $\mathrm{T} = 0.7$.

$\mathcal{S}$	WeakAXp?	AXp?	$\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \ominus \mid (\mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}}))$	WeakPAXp?	PAXp?	LmPAXp?	MinPAXp?
$\{1, 2, 3\}$	Yes	Yes	$1 \geq \mathrm{T}$	Yes	No	No	No
$\{1, 2\}$	No	No	$3/4 = 0.75 \geq \mathrm{T}$	Yes	Yes	Yes	No
$\{1, 3\}$	No	No	$4/5 = 0.8 \geq \mathrm{T}$	Yes	No	No	No
$\{2, 3\}$	No	–	$3/4 = 0.75 \geq \mathrm{T}$	Yes	No	No	No
$\{1\}$	No	–	$12/20 = 0.6 \leq \mathrm{T}$	No	No	No	No
$\{2\}$	No	–	$9/16 = 0.5625 \leq \mathrm{T}$	No	No	No	No
$\{3\}$	No	–	$14/20 = 0.7 \geq \mathrm{T}$	Yes	Yes	Yes	Yes

performed by encoding each neuron in the network with a constraint of the form:

$$y \leftrightarrow \sum_{i=1}^{n} w_i \cdot l_i \geq b, \quad y \in \{0, 1\} \tag{7}$$

where l_i are Boolean variables, w_i and b are integer variables[2] and $(2 \cdot y - 1)$ is the output of the (internal) neuron. Note that (7) can be easily transformed into a *reified cardinality constraint* with unary coefficients, and thus be encoded with sequential encoders [53, 54] in clausal form, which yields a CNF representation of the BNN.

Besides, (7) can be rewritten into two PB constraints:

$$\begin{array}{lll} \sum_{i=1}^{n} w_i l_i + (b + N) \cdot \neg y \geq b & \text{for} & y \rightarrow \sum_{i=1}^{n} w_i l_i \geq b \\ \sum_{i=1}^{n} w_i l_i + (b - N) \cdot y \leq b & \text{for} & \neg y \rightarrow \sum_{i=1}^{n} w_i l_i < b \end{array}$$

with $N = \sum_{i=1}^{n} |w_i|$.

Next, we need to encode the output layer of the neural network, i.e. $\kappa(\mathbf{x})$. Let s_j be a Boolean variable representing class $c_j \in \mathcal{K}$, W_j^i is a tensor of weights and b_j is a bias of a neuron j in the last layer of the BNN. Hence, a target class c_j is picked ($s_j = 1$) iff,

$$\bigwedge\nolimits_{k \neq j} y_k \leftrightarrow \Big((W_k^i - W_j^i) \cdot l_i \geq (b_k - b_j)\Big) \wedge \Big(s_j \leftrightarrow \bigwedge\nolimits_{k \neq j} \neg y_k\Big)$$

RF Encoding. We adapt the propositional encoding of RFs proposed in [21] to encode $\kappa(\mathbf{x})$ entails the prediction of input $\mathbf{v}$, in order to be able to compute the number of models satisfying this entailment. Concretely, we need to encode the RF structure (i.e. trees and feature-domain encodings) and the prediction function $\kappa(\mathbf{x})$ (i.e. majority vote of the ensemble model) into a SAT problem. As results, each tree $\mathcal{T}_i$ of the model is encoded as a set of implications expressing the paths $\mathcal{P}_i$ of $\mathcal{T}_i$, as follow: $\bigwedge_{P \in \mathcal{P}_i} \Big(\bigwedge_{l \in \Lambda(P)} l \leftrightarrow p_{ij}\Big)$, where $\Lambda(P)$ is the set of literals of path P and p_{ij} represents the class label c_j of P, such that $p_{ij} = 1$ if the class c_j is predicted when P is consistent. Next, we outline the encoding of majority vote for $\kappa(\mathbf{x})$. Then, the idea is to enforce the number of votes for the target class c_j to be the largest one for $\kappa(\mathbf{x})$:

$$\bigwedge\nolimits_{k \neq j} \sum\nolimits_{i=1}^{n} p_{ij} + \sum\nolimits_{i=1}^{n} \neg p_{ik} + b_k \geq n + 1$$

where b_k serves to preserve the lexicographic order between c_j and c_k, in case of ties. Therefore, $b_k = 1$ iff $j > k$; and when $j < k$ the cardinality constraint enforces the inequality $\big(\sum_{i=1}^{n} p_{ij} > \sum_{i=1}^{n} \neg p_{ik}\big)$, then RHS equals to $n + 1$.

[2] For more details on how the inequality is calculated (coefficients, bound, etc) the reader can refer to [45] (Sec. 4) or [46] (Sec. 3.1).

4 Approximate Locally Minimal Probabilistic Explanations

We are interested in (approximately) computing LmPAXp's for complex ML models, including tree ensembles and binarized neural networks. Therefore, the difficulty is to compute or estimate the probability of the logical formula representing the model to be satisfiable. Clearly, applying exact model counting for large propositional formulas will be infeasible in practice. We solve this problem using either approximate model counting or sampling, with strong probably approximately correct (PAC) guarantees. A procedure implementing one of these approaches will serve to check whether a candidate set of features $\mathcal{X}$ is a weak PAXp using model counting or sampling technique. Additionally, a linear search algorithm is proposed to compute LmPAXp's which instruments the predicate WeakPAXp checker subroutine.

4.1 Approximate Model Counting

The approximate model counter serves as oracle in the linear search analysis to compute the number of points $\mathbf{x} \in \mathbb{F}$, such that the condition $\kappa(\mathbf{x}) = \kappa(\mathbf{v})$ and $\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}}$ holds. We formulate this problem as a SAT (or pseudo-Boolean) formula (as described earlier in Section 3) and instrument an oracle call to a PAC counter ApproxMC[3] [5, 55, 56, 42] (resp. ApproxMCPB [63, 64] for PB formulas) that returns an (ϵ, δ)-approximate number of solutions η^* of the input formula, such that

$$\Pr\left[\frac{\eta}{1+\epsilon} \leq \eta^* \leq (1+\epsilon) \times \eta\right] \geq 1 - \delta$$

where η is the exact number of solutions for the input formula, $\epsilon > 0$ is the tolerance and $(1 - \delta)$ is the confidence of the PAC probability. Observe that η corresponds exactly to the numerator $|\{\mathbf{x} \in \mathbb{F} : \kappa(\mathbf{x}) = c \wedge (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}|$ of the conditional probability $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = c \,|\, \mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})$ in (4), and the bounded approximation η^* is subsequently a PAC approximation of $|\{\mathbf{x} \in \mathbb{F} : \kappa(\mathbf{x}) = c \wedge (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}|$. Additionally, it is plain that $|\{\mathbf{x} \in \mathbb{F} : (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}| \geq 1$ (equals 1 when all features are fixed to values of $\mathbf{v}$, i.e. $\mathcal{X} = \mathcal{F}$), hence we have a probability bounded approximation $\eta^* / |\{\mathbf{x} \in \mathbb{F} : \kappa(\mathbf{x}) = c \wedge (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}|$ with a tolerance ϵ and a confidence $(1 - \delta)$,

$$\frac{p}{1+\epsilon} \leq \frac{\eta^*}{|\{\mathbf{x} \in \mathbb{F} : \kappa(\mathbf{x}) = c \wedge (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}|} \leq (1+\epsilon) \times p$$

where p denotes $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = c \,|\, \mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})$.

[3] ApproxMC is a state-of-the-art approximate model counter, that scales for large problem instances and provides rigorous approximation guarantees.

Algorithm 1 Deletion-based method for computing LmPAXp

Input: Parameters: $\mathcal{E}$, T; Hyperparameters ϵ, δ
Output: LmPAXp $\mathcal{S}$ s.t. $\Pr[\Pr_{\mathbf{x}}(\mathcal{S}) \geq \mathrm{T} - \epsilon] \geq 1 - \delta$

1: **procedure** DelLmPAXp($\mathcal{E}$, T; ϵ, δ)
2: $\mathcal{S} \leftarrow \mathcal{F}$
3: $\Phi \leftarrow \mathsf{encode}(\mathcal{E})$
4: **for** $i \in \mathcal{F}$ **do**
5: $mc \leftarrow \mathsf{approxCount}(Sol(\Phi_{|\mathcal{S}\setminus\{i\}}), \epsilon, \frac{\delta}{m})$
6: $pr \leftarrow {}^{mc}/_{\prod_{j\in\mathcal{F}\setminus(\mathcal{S}\setminus\{i\})} ||\mathbb{D}_j||}$
7: **if** $pr \geq \mathrm{T}$ **then** ▷ WeakPAXp($\mathbf{S}$)?
8: $\mathcal{S} \leftarrow \mathcal{S} \setminus \{i\}$
9: **return** $\mathcal{S}$

4.2 Monte-Carlo Sampling

Our second proposed approach is Monte Carlo sampling (MC-sampling) over $\mathbf{x}_{\mathcal{S}} \sim \mathbb{F}$ (uniformly sample from $\mathbb{F}$ restricted to variables of $\mathcal{F} \setminus \mathcal{S}$ such that $\bigwedge_{i\in\mathcal{S}} x_i = v_i$) and test κ on each sample as input — this will give the count of samples that are in the same class as input $\mathbf{v}$ (i.e. $\kappa(\mathbf{x}) = \kappa(\mathbf{v})$). In the same vein as for approximate model counting method, we propose an (ϵ, δ)-approximation solution. Subsequently, The number of samples to generate is identified by the standard Chernoff bounds [44] expressed with the parameters of tolerance ϵ and confidence δ. Concretely, let X_i be a 0–1 random variable denoting the result of the trial with sample $\mathbf{z}_i$, where $X_i = 1$ iff $\mathbb{1}_{(\kappa(\mathbf{z}_i)=\kappa(\mathbf{v}))}(\mathbf{z}_i) = 1$ and $X_i = 0$ otherwise. Let X be the random variable denoting the number of trials in $X_1, X_2, \ldots, X_N$ for which $\mathbb{1}_{(\kappa(\mathbf{z}_i)=\kappa(\mathbf{v}))}$ returns 1. Then,

Proposition 1 (Hoeffding [61])**.** Given N independent 0–1 random variables X_i, $X = \frac{1}{N}\sum_{i=1}^{N} X_i$, the expected value $\mu = \mathbb{E}(X)$ and $\epsilon > 0$, $\Pr(\mid X - \mu \mid \geq \epsilon) \leq 2e^{-2\epsilon^2 N}$.

Let $\delta > 0$ such that $2e^{-2\epsilon^2 N} \leq \delta$, then we have, $N \geq \frac{1}{2\epsilon^2}\log(\frac{2}{\delta})$.

Note that N is the minimum number of samples that needed to be (randomly) drawn in order to guarantee the Hoeffding bounds on μ,

$$\Pr[X - \epsilon \leq \mu \leq X + \epsilon] \geq 1 - \delta$$

Clearly, we can estimate the probability of an explanation through sampling by applying the Hoeffding bounds to empirically estimate the mean X in N trial. Hence, the estimate μ represents the (ϵ, δ)-approximation probability of the explanation precision $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \kappa(\mathbf{v}) \,|\, \mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})$.

Observe that for MC-sampling method, we have an additive bound (i.e. $\pm\epsilon$), whilst we have a multiplicative bound (i.e. a factor of $(1 + \epsilon)$) with approximate model counting. Clearly, multiplicative bound probability is a stronger/tighter bound, subsequently provides a more accurate approximation of the true probability of the explanation. Nevertheless, as shown in the results (see Section 5), for small ϵ and δ that is not close to zero (e.g. $\delta \simeq 0.05$), the MC-sampling technique shows similar precisions as approximate model counting and notably significantly faster to compute.

4.3 Computing Locally-Minimal PAXp's

Algorithm 1 depicts a (deletion-based) linear search method for computing a locally-minimal PAXp. As shown, to compute one LmPAXp $\mathcal{S}$, one can starts from an initial set $\mathcal{F} = \{1, \ldots, m\}$ containing all features of the data or immediately from an AXp $\mathcal{X} \subset \mathcal{F}$ and iteratively removes least contributing features while it is safe to do so, i.e. while Eq. (4) holds for the resulting set. Concretely, the procedure computes for $\mathcal{S}$ an approximation of the number of solutions that admits the updated subformula $\Phi \wedge \bigwedge_{j\in\mathcal{S}}(x_j = v_j)$ (or denoted $\Phi_{|\mathcal{S}}$ representing the classifier restricted to $\mathcal{S}$) and subsequently calculates the probability $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \kappa(\mathbf{x}) \,|\, \mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}})$ and checks if it is still greater than T. The model counting approach and Monte-Carlo sampling method are implemented in the procedure approxCount — the choice for the approach to use is configured in the parameters of the function. Moreover, we implement an heuristic using Monte-Carlo sampling approach to order the features from least to highest heuristic score, which allows in practice to obtain smaller probabilistic explanations. Note that for FFAXp explanations, one can apply the FFA scores in the heuristic order of the features.

Observe that features not belonging to an AXp do not contribute in the decision of $\kappa(\mathbf{v})$ and thus can be safely removed at the initialisation step, which allows us to improve the performance of Algorithm 1, i.e. initialize $\mathcal{S}$ to some AXp $\mathcal{X} \subseteq \mathcal{F}$. Our empirical results showcase that the fractions between the AXp's size and the resulting locally-minimal AXp's are significantly large.

Note that besides linear search algorithms for solving function problems in propositional logic and constraint programming [40, 39], QuickXplain [30] is another alternative approach that could be considered for this problem.

5 Experiments

This section summarizes the empirical assessment of computing Probabilistic Abductive explanations for the case study of RF and BNN models trained on some of the widely studied datasets. Moreover, we assess our approach with the baseline method for computing minimal probabilistic approach for decision trees (DTs) [25]. This comparison allows us to validate the accuracy and effectiveness of our approximation method with the exact method for a scalable family of classifiers.

Experimental setup. The experiments are conducted on Intel Core i5-10500 3.1GHz CPU with 16GByte RAM running Ubuntu 22.04. LTS. A time limit for each single call to the approximate counter is fixed to 120 seconds and a general time limit for delivering an explanation is set to 600 seconds; whilst the memory limit is set to 4 GByte.

Benchmarks. The assessment of RFs is performed on a selection of 17 publicly available datasets, which originate from UCI Machine Learning Repository [58] and Penn Machine Learning Benchmarks [47]. Benchmarks comprise binary and multidimensional classification datasets and include binary or/and categorical datasets. (Categorical features are encoded into bit-vectors and handled as a group of attributes representing their original features when computing the explanations.) We RFs with 100 trees and used 80% of the dataset instances (20% used for test data). For assessing explanation tools, we randomly picked fractions of the dataset, depending on the dataset size. Besides, the assessment of BNNs is performed on image datasets of MNIST digits [32]. First, we binarized the data (convert pixel values into 0 or 1) and generated 7 binary class datasets from the original multi-class (from 0 to 9) dataset. Namely, each resulting binary dataset contains images of 2 labels c_1 *versus* c_2 (e.g. mnist-0vs8 comprises samples of class 0 and 8). We considered different tunings of the training parameters so that we obtain the best training and test accuracy of each benchmark. The precision threshold T of LmPAXp is fixed to 95% for all considered benchmarks in RFs and 99% for BNNs. Hence, to guarantee high probabilistic precision

of the explanations we implement the ApproxMC metric for BNNs; whilst for a better scalability and without loosing in accuracy, we apply the MC-Sampling metric for RFs. Note that, our observation on preliminary results demonstrate that for $\mathrm{T} \leq 0.97$ both metrics (sampling and model counting) return similar results. Lastly, for assessment on DTs we re-used the exact benchmarks (i.e. datasets and models) considered in [25].

Prototypes implementation. We developed a prototype computing LmPXAp's for RFs in Python[4] on top of the SAT-based reasoner RFxp [21] for RFs. Furthermore, PySAT [16] is used to instrument incremental SAT oracle calls, and the oracles ApproxMC[5] [5, 55, 56, 42] and ApproxMCPB[6][63, 64] to instrument (approximate) model counting calls on SAT and pseudo-Boolean formulas. Besides, we reused the BNN encoding of [29][7] to implement both SAT and pseudo-Boolean -based encodings for our formulation of computing explanations. Similarly to RF encoding, we used PySAT to generate BNN SAT encoding, and we used the Python package PyPBLib [50] of PBLib [49], which is integrated in PySAT, to generate the BNN pseudo-Boolean encoding. (Note that PyPBLib/PBLib by design provides efficient clausal form encodings of pseudo-Boolean formulas. Accordingly, we used it on purpose to generate the pseudo-Boolean constraints and convert them into CNF formulas when SAT encoding is selected.) Moreover, Roundingsat[8] [10] oracle is instrumented to reason on PB formulas when computing AXp's for BNNs. We underline that our observations on preliminary assessments, and also as pointed out in [29], PB encoding and Roundingsat oracle shows (slightly) better performances than SAT oracles. As a result, we solely report the experimental results of the PB encodings for BNN benchmarks.

Baseline. In order to validate our approach in terms of explanation accuracy and size, we used as a baseline the dedicated approach for computing MinPAXp's [25] for DTs.[9] Besides, we underline that the aim of this paper is not to compete with the exact approach tailored for DTs in [24, 25], which has been shown efficient in practice, but to propose a solution for more complex ML models for which exact methods are unable to scale. As can be seen from Table 2, our method based on MC-Sampling shows (almost) comparable results with [25], in terms of explanation length and precision. Furthermore, although the computation of MinPAXp's shows better runtimes, we observe that LmPAXp's runtimes are also very small (i.e. less than 1s (resp. 2s) for 6/8 (2/8)). Hence, one can argue that our *general* solution is competitive with model specific techniques and these additional results empirically validate explanations faithfulness delivered by our method.

MC-Sampling. Table 3 summarizes the results of assessing the probabilistic explanation succinctness of RFs using MC-Sampling technique. Notably, Table 3 reports the average lengths of LmPAXp's *versus* with AXp's, and FFAXp's *versus* LmFFAXp's computed with Algorithm 1 coupled with MC-sampling. As can be observed from Table 3, with a few exceptions (2 (binary) datasets out of 17, i.e.

[4] The source code is available at https://github.com/izzayacine/praxp
[5] https://github.com/meelgroup/approxmc
[6] https://github.com/meelgroup/approxmcpb
[7] https://github.com/jia-kai/eevbnn
[8] Roundingsat is a PB solver augmented with the LP solver SoPlex [57].
[9] Note that a comparison with the greedy algorithm of [3] designed for DTs, would not be conclusive, since it is not an exact method. Moreover, we underscore that [3] did not validate their algorithm with the baseline exact (SMT-based) approach of [24, 25], under the claim that the source code was not yet available at the time they were writing their paper, however so far, they did not make their implementation publicly available.

Table 2: Assessing LmPAXp's for DTs and compare with *minimum* PAXp's [25]. Threshold T was fixed to 0.9. Columns **m** shows the number of features and **#S** the number of tested instance for each dataset. Columns **Len** and **Prec** report, resp., the average length and accuracy of computed LmPAXp's (resp. MinPAXp) and **Time** shows the average runtime for computing such explanations.

Dataset			MinPAXp			LmPAXp		
	m	**#S**	**Len**	**Prec**	**Time**	**Len**	**Prec**	**Time**
adult	12	4848	5.6	94.6	4.64	5.7	94.79	0.17
dermatology	34	292	4.0	98.8	0.35	4.0	98.86	0.13
kr-vs-kp	36	2556	3.2	95.4	0.92	3.3	95.40	0.21
letter	16	14934	7.7	97.7	16.35	7.7	97.69	0.25
soybean	35	498	6.1	98.1	0.94	6.1	98.21	0.22
spambase	57	3368	2.4	92.4	2.15	2.6	93.12	0.93
factors	216	1595	5.3	98.30	1.64	5.3	98.28	1.28
texture	40	4378	5.4	98.5	2.20	5.3	98.90	2.06

mofn_3_7_10 and *parity5+5*) LmPAXp provides shorter explanations for all tested datasets — this is illustrated for example, in *ionosphere* and *german* datasets the average size of LmPAXp's is, resp., 86.6% and 73.9% smaller than AXp's, with an average precision of 0.97 on all tested samples. Moreover, for the majority of the datasets the average precision is 0.97 or higher, in total 16 out 17 datasets (i.e. 8, 5 and 3 of them, resp. show an average precision of 0.97, 0.98 and 0.99) and 1 dataset has an average precision of 0.96. Besides, according to our expectations, the average runtimes for MC-sampling are small — the minimum average runtime reported is 0.13 seconds, while the maximum does not exceed 4.8 seconds. Unsurprisingly FFAXp's are even larger that AXp's, however we observe that LmPFFAXp average length's are (almost) similar to LmPAXp's. As a result, one can remark a significant gain of size for LmPFFAXp's, e.g. for *agaricus* the average length of FFAXp is 20 (two times the average length of AXp's) and the average length of LmPFFAXp's is 3.4 (tightly close to the average size of LmPAXp's, which is 3.3), i.e. a gap of 83%; and for *soybean* we have LmPFFAXp average size that represents only 26% of the initial FFAXp average size.

ApproxMCounting. Recall that for high accuracy of probabilistic explanations, ApproxMC is more faithful offering higher probabilistic guarantees than MC-Sampling. We evaluate our ApproxMC technique for computing LmPAXp's/LmPFFAXp's on a more challenging family of classifiers than MC-Sampling assessment. The aim is to showcase that for classifiers like BNNs trained on image data, we are able to obtain more succinct formal probabilistic explanations than formal explanations, with high probability of precision. The counterpart is the runtime of ApproxMC that is usually higher than MC-Sampling, nevertheless our results show it is possible to reduce significantly the size of AXp's in a reasonable time.

The obtained results on BNNs, summarized in Table 4, show significant size reduction in the explanations of LmPAXp compared to AXp while offering a very high precision $\mathrm{T} = 0.99$. We observe a difference of average length of 34.5% up to 68.2% between LmPAXp's and AXp's, e.g. the average number of pixels for AXp in *mnist-0vs8* is 29, whereas for LmPAXp it is 10 pixels, which represents 33.6% of its average AXp size. Similarly as RFs, we observe a large gain of size from FFAXp's to LmFFAp's. In fact by construction, FFAXp's are larger than AXp's, in contrast we observe that the resulting LmFFAXp's are always as succinct as LmPAXp's, even though the explanation sets could be very different, since the initial sets of features to inspect are different and the order for analyzing the features are also different (i.e. for LmPFFAXp's we apply the ffa score and for AXp's either lexicographic or MC-Sampling score). In terms of runtimes, the averages reported for LmPAXp's vary from

Table 3: Detailed performance evaluation of computing LmPAXp and LmPFFAXp explanations for RFs. The table shows results for 17 datasets containing binary and categorical data. The maximum tolerance was fixed to 0.05, i.e. $\mathrm{T} = 0.95$. Column **%** reports the average percentage of features preserved in the probabilistic explanations. (Remaining columns keep the same meaning as in Table 2.)

Dataset	(m, K)	AXp		LmPAXp				FFAXp		LmPFFAXp			
		Len	Time	Len	%	Prec	Time	Len	Time	Len	%	Prec	Time
adult	(12 2)	5.6	0.19	3.2	57	0.97	0.99	6.0	0.04	5.0	83	0.99	0.67
agaricus	(22 2)	10.0	0.06	3.3	33	0.96	3.78	20.0	24.66	3.4	17	0.97	16.81
compas	(11 2)	5.6	0.12	3.6	64	0.97	0.77	10.0	0.37	3.9	39	0.97	2.75
german	(21 2)	12.3	0.36	3.2	26	0.97	5.40	21.0	155.64	3.4	16	0.97	16.01
heart-c	(13 2)	5.5	0.07	3.0	55	0.97	0.39	13.0	1.99	2.9	22	0.97	2.13
ionosphere	(34 2)	21.5	0.06	2.9	13	0.96	6.59	33.0	13.17	2.9	9	0.97	14.68
kr_vs_kp	(36 2)	8.1	0.08	3.5	43	0.98	0.84	33.0	3.58	3.3	10	0.99	15.46
lending	(9 2)	2.2	0.10	1.7	77	0.99	0.15	2.0	0.03	1.7	85	0.99	0.12
mofn_3_7_10	(10 2)	3.1	0.06	2.8	90	0.99	0.13	5.0	0.01	3.4	68	0.99	0.29
mushroom	(22 2)	9.2	0.07	4.2	46	0.97	2.30	20.0	4.72	4.3	22	0.97	11.16
parity5+5	(10 2)	6.8	0.23	6.4	94	0.99	0.38	9.0	0.03	6.7	74	0.99	0.82
recidivism	(15 2)	7.9	0.48	5.8	73	0.97	1.75	15.0	0.43	5.7	38	0.97	8.49
segmentation	(19 7)	8.1	0.39	4.9	60	0.98	0.76	17.0	10.42	4.9	29	0.98	4.11
soybean	(35 18)	15.6	2.05	8.9	57	0.97	4.07	34.0	230.38	8.7	26	0.97	24.36
tic_tac_toe	(9 2)	5.0	0.09	3.3	66	0.97	0.58	9.0	0.03	3.4	38	0.97	2.17
twonorm	(20 2)	10.4	0.05	4.4	42	0.97	1.35	15.0	0.04	8.1	54	0.98	1.97
vote	(16 2)	5.6	0.07	2.6	46	0.98	1.16	16.0	1.26	2.5	16	0.98	10.65

Table 4: Detailed performance evaluation of computing locally-minimal PAXp for BNNs trained on (binarized) MNIST datasets. The maximum tolerance is fixed to 0.01, i.e. $\mathrm{T} = 0.99$. Column **#TO** reports the number timeouts reported over 100 tests for Algorithm 1.

Dataset	AXp		LmPAXp				FFAXp		LmFFAXp			
	Len	Time	Len	%	Time	#TO	Len	Time	Len	%	Time	#TO
mnist-0vs8	28.8	0.51	10.1	35	345.24	11	42.68	1.40	10.2	24	443.95	12
mnist-1vs6	25.6	0.52	9.0	35	125.69	0	44.24	1.79	9.0	20	281.40	0
mnist-2vs6	18.6	0.47	12.0	65	27.76	0	27.62	0.18	12.2	44	75.50	0
mnist-1vs8	29.3	0.50	12.5	43	150.10	0	43.78	1.51	12.7	29	279.08	1
mnist-2vs8	29.6	0.48	12.6	43	169.04	0	47.66	1.57	15.2	26	312.76	0
mnist-3vs8	29.7	0.48	11.8	40	118.16	0	43.14	0.78	13.3	31	249.88	0
mnist-4vs9	26.1	0.47	9.8	38	87.81	0	38.32	0.67	9.7	25	182.84	0

27.76 seconds to 345.24 seconds (avg. 146.25 seconds), with 6 out of 7 BNNs that successfully returned an explanation without exceeding the timeout fixed for ApproxMC oracle, however *mnist-0vs8* reports 11% timeouts. Furthermore, for LmPFFAXp's we remark that runtimes are slightly higher than LmPAXp's which is not surprising since the algorithm performs more iterations in this case compared to FFAXp's where the initial set of features AXp to inspect is smaller. Also, we observe that out of 7 BNNs 5 show average runtimes less than 300 seconds (5 minutes) and a total average of 260.77 seconds.

Given the above discussion, we can conclude that our solutions for computing locally-minimal formal explanations for complex models enabling logical representation, such as tree ensembles, binarized neural networks, decision lists, etc, are effective and offer succinct explanations with probabilistic guarantees of precision.

6 Conclusions

Formal explainability witnessed significant progress in recent years. However, some challenges remain, including the size of explanations. Probabilistic abductive explanations (PAXp's) represent a solution to curb explanation size. Unfortunately, their exact computation is impractical. An alternative to PAXp's are locally minimal probabilistic abductive explanations (LmPAXp's), which recent work showed, for some families of classifiers, that LmPAXp's match in practice PAXp's. Motivated by these results, this paper proposes two techniques for approximating the computation of LmPAXps for complex ML models. The experimental results reveal two possible use-cases, one for each of the proposed methods. In application domains where probabilistic formal explanations suffice, i.e. 90% to 97% precision, the paper proposes a novel method based on MC-sampling, which scales to *any* classifier. Alternatively, in domains where explanations with high accuracy (i.e. around 99%) are of interest, the paper proposes another novel method based on approximate model counting. Future work will consider the use of approximate model counting for families of classifiers beyond boolean domains. This will require integration of approximate model counting with SMT reasoners.

Acknowledgements

This work was supported in part by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, by the starting funds of ICREA, and by the project PID2022-139835NB-C22 funded by MCIN/AEI/10.13039/501100011033/FEDER, UE.

References

[1] M. Arenas, P. Barceló, M. Romero, and B. Subercaseaux. On computing probabilistic explanations for decision trees. In *NeurIPS*, 2022.

[2] G. Biradar, Y. Izza, E. Lobo, V. Viswanathan, and Y. Zick. Axiomatic aggregations of abductive explanations. In M. J. Wooldridge, J. G. Dy, and S. Natarajan, editors, *AAAI*, pages 11096–11104, 2024.

[3] L. Bounia and F. Koriche. Approximating probabilistic explanations via supermodular minimization. In *UAI*, 2023.

[4] L. Breiman. Random forests. *Mach. Learn.*, (1):5–32, 2001. URL https://doi.org/10.1023/A:1010933404324.

[5] S. Chakraborty, K. S. Meel, and M. Y. Vardi. Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In *IJCAI*, pages 3569–3576, 2016.

[6] C. Cheng, G. Nührenberg, C. Huang, and H. Ruess. Verification of binarized neural networks via inter-neuron factoring - (short paper). In *VSTTE*, volume 11294, pages 279–290, 2018.

[7] M. C. Cooper and J. Marques-Silva. On the tractability of explaining decisions of classifiers. In *CP*, pages 21:1–21:18, 2021.

[8] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training neural networks with weights and activations constrained to +1 or -1. *CoRR*, 2016.

[9] A. Darwiche. Logic for explainable AI. In *LICS*, pages 1–11, 2023.

[10] J. Elffers and J. Nordström. Divide and conquer: Towards faster pseudo-boolean solving. In *IJCAI*, pages 1291–1299. ijcai.org, 2018.

[11] J. Feng and Z. Zhou. AutoEncoder by forest. In *AAAI*, pages 2967–2973, 2018.

[12] W. Gao and Z. Zhou. Towards convergence rate analysis of random forests for classification. In *NeurIPS*, page 103788, 2020.

[13] X. Huang and J. Marques-Silva. On the failings of shapley values for explainability. *Int. J. Approx. Reason.*, page 109112, 2024.

[14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *NeurIPS*, pages 4107–4115, 2016.

[15] A. Ignatiev. Towards trustable explainable AI. In *IJCAI*, pages 5154–5158, 2020.

[16] A. Ignatiev, A. Morgado, and J. Marques-Silva. PySAT: A python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018.

[17] A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519, 2019.

[18] A. Ignatiev, N. Narodytska, and J. Marques-Silva. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509, 2019. URL http://arxiv.org/abs/1907.02509.

[19] A. Ignatiev, Y. Izza, P. J. Stuckey, and J. Marques-Silva. Using maxsat for efficient explanations of tree ensembles. In *AAAI*, pages 3776–3785. AAAI Press, 2022.

[20] A. Ignatiev, Y. Izza, P. J. Stuckey, and J. Marques-Silva. Using MaxSAT for efficient explanations of tree ensembles. In *AAAI*, pages 3776–3785, 2022.

[21] Y. Izza and J. Marques-Silva. On explaining random forests with SAT. In *IJCAI*, pages 2584–2591, 2021.

[22] Y. Izza and J. Marques-Silva. Enigma - kr. volume 3495 of *CEUR Workshop Proceedings*, pages 75–86. CEUR-WS.org, 2023.

[23] Y. Izza, A. Ignatiev, N. Narodytska, M. C. Cooper, and J. Marques-Silva. Efficient explanations with relevant sets. *CoRR*, abs/2106.00546, 2021. URL https://arxiv.org/abs/2106.00546.

[24] Y. Izza, A. Ignatiev, N. Narodytska, M. C. Cooper, and J. Marques-Silva. Provably precise, succinct and efficient explanations for decision trees. *CoRR*, abs/2205.09569, 2022.

[25] Y. Izza, X. Huang, A. Ignatiev, N. Narodytska, M. C. Cooper, and J. Marques-Silva. On computing probabilistic abductive explanations. *Int. J. Approx. Reason.*, 159:108939, 2023.

[26] Y. Izza, K. S. Meel, and J. Marques-Silva. Locally-minimal probabilistic explanations. *CoRR*, abs/2312.11831, 2023.

[27] Y. Izza, X. Huang, A. Morgado, J. Planes, A. Ignatiev, and J. Marques-Silva. Distance-restricted explanations: Theoretical underpinnings & efficient implementation. In *KR*, 2024.

[28] Y. Izza, A. Ignatiev, P. J. Stuckey, and J. Marques-Silva. Delivering inflated explanations. In *AAAI*, pages 12744–12753, 2024.

[29] K. Jia and M. C. Rinard. Efficient exact verification of binarized neural networks. In *NeurIPS*, 2020.

[30] U. Junker. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In *AAAI*, pages 167–172, 2004.

[31] J. Kung, D. C. Zhang, G. S. van der Wal, S. M. Chai, and S. Mukhopadhyay. Efficient object detection using embedded binarized neural networks. *J. Signal Process. Syst.*, 90(6):877–890, 2018.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[33] O. Letoffe, X. Huang, N. Asher, and J. Marques-Silva. From SHAP scores to feature importance scores. *CoRR*, 2024. URL https://doi.org/10.48550/arXiv.2405.11766.

[34] J. Marques-Silva. Disproving XAI myths with formal methods - initial results. In *ICECCS*, pages 12–21, 2023.

[35] J. Marques-Silva. Logic-based explainability: Past, present & future. *CoRR*, 2024. URL https://doi.org/10.48550/arXiv.2406.11873.

[36] J. Marques-Silva and X. Huang. Explainability is not a game. *Commun. ACM*, (7):66–75, 2024. URL https://doi.org/10.1145/3635301.

[37] J. Marques-Silva and A. Ignatiev. Delivering trustworthy AI through formal XAI. In *AAAI*, pages 12342–12350, 2022.

[38] J. Marques-Silva and A. Ignatiev. No silver bullet: interpretable ML models must be explained. *Frontiers Artif. Intell.*, 6, 2023. doi: 10.3389/FRAI.2023.1128212. URL https://doi.org/10.3389/frai.2023.1128212.

[39] J. Marques-Silva, M. Janota, and A. Belov. Minimal sets over monotone predicates in boolean formulae. In *CAV*, pages 592–607, 2013.

[40] J. Marques-Silva, M. Janota, and C. Mencía. Minimal sets on propositional formulae. problems and reductions. *Artif. Intell.*, 252:22–50, 2017. doi: 10.1016/j.artint.2017.07.005. URL https://doi.org/10.1016/j.artint.2017.07.005.

[41] B. McDanel, S. Teerapittayanon, and H. T. Kung. Embedded binarized neural networks. In *EWSN*, pages 168–173, 2017.

[42] K. S. Meel and S. Akshay. Sparse hashing for scalable approximate model counting: Theory and practice. In *LICS*, pages 728–741, 2020.

[43] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81–97, 1956.

[44] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2nd edition, 2017.

[45] N. Narodytska, S. P. Kasiviswanathan, L. Ryzhyk, M. Sagiv, and T. Walsh. Verifying properties of binarized deep neural networks. In *AAAI*, pages 6615–6624, 2018.

[46] N. Narodytska, H. Zhang, A. Gupta, and T. Walsh. In search for a SAT-friendly binarized neural network architecture. In *ICLR*, 2020.

[47] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, (36):1–13, 2017.

[48] F. Pedregosa and et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[49] T. Philipp and P. Steinke. Pblib – a library for encoding pseudo-boolean constraints into cnf. In *SAT*, pages 9–16. 2015.

[50] pypblib. PyPBLib: Python Pseudo-Boolean library. http://ulog.udl.cat/static/doc/pypblib/html/index.html, 2023.

[51] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, pages 1527–1535, 2018.

[52] A. Shih, A. Choi, and A. Darwiche. A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, pages 5103–5111, 2018.

[53] C. Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *CP*, pages 827–831, 2005.

[54] C. Sinz and E. Dieringer. Dpvis - A tool to visualize the structure of SAT instances. In *SAT*, pages 257–268, 2005.

[55] M. Soos and K. S. Meel. BIRD: engineering an efficient CNF-XOR SAT solver and its applications to approximate model counting. In *AAAI*, pages 1592–1599, 2019.

[56] M. Soos, S. Gocht, and K. S. Meel. Tinted, detached, and lazy CNF-XOR solving and its applications to counting and sampling. In *CAV*, pages 463–484, 2020.

[57] SoPlex. Sequential object-oriented simplex. https://soplex.zib.de, 2023.

[58] UCI. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml.

[59] S. Wäldchen, J. MacDonald, S. Hauch, and G. Kutyniok. The computational complexity of understanding binary classifier decisions. *J. Artif. Intell. Res.*, 70:351–387, 2021.

[60] E. Wang, P. Khosravi, and G. V. den Broeck. Probabilistic Sufficient Explanations. In *IJCAI*, pages 3082–3088, 2021.

[61] M. Wooldridge. Computational aspects of cooperative game theory. In *KES-AMSTA*, 2011.

[62] M. Wu, H. Wu, and C. W. Barrett. VeriX: Towards verified explainability of deep neural networks. In *NeurIPS*, 2023.

[63] J. Yang and K. S. Meel. Engineering an efficient PB-XOR solver. In *CP*, pages 58:1–58:20, 2021.

[64] J. Yang and K. S. Meel. Rounding meets approximate model counting. In *CAV*, pages 132–162, 2023.

[65] L. Yang, X. Wu, Y. Jiang, and Z. Zhou. Multi-label learning with deep forest. In *ECAI*, pages 1634–1641, 2020.

[66] J. Yu, A. Ignatiev, and P. J. Stuckey. On formal feature attribution and its approximation. *CoRR*, abs/2307.03380, 2023.

[67] J. Yu, G. Farr, A. Ignatiev, and P. J. Stuckey. Anytime approximate formal feature attribution. In *SAT*, pages 30:1–30:23, 2024.

[68] Y. Zhang, J. Zhou, W. Zheng, J. Feng, L. Li, Z. Liu, M. Li, Z. Zhang, C. Chen, X. Li, Y. A. Qi, and Z. Zhou. Distributed deep forest and its application to automatic detection of cash-out fraud. *ACM Trans. Intell. Syst. Technol.*, 10(5):55:1–55:19, 2019.

[69] Z. Zhou and J. Feng. Deep forest: Towards an alternative to deep neural networks. In *IJCAI*, pages 3553–3559, 2017.