Learning Optimal Decision Trees with MaxSAT and its Integration in AdaBoost

Hao Hu[1*], **Mohamed Siala**[1], **Emmanuel Hebrard**[1], **Marie-José Huguet**[1]
LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France[1]
{hhu, siala, hebrard, huguet}@laas.fr

Abstract

Recently, several exact methods to compute decision trees have been introduced. On the one hand, these approaches can find optimal trees for various objective functions including total size, depth or accuracy on the training set and therefore. On the other hand, these methods are not yet widely used in practice and classic heuristics are often still the methods of choice. In this paper we show how the SAT model proposed by [Narodytska *et al.*, 2018] can be lifted to a MaxSAT approach, making it much more practically relevant. In particular, it scales to much larger data sets; the objective function can easily be adapted to take into account combinations of size, depth and accuracy on the training set; and the fine-grained control of the objective function it offers makes it particularly well suited for boosting. Our experiments show promising results. In particular, we show that the prediction quality of our approach often exceeds state of-the-art heuristic methods. We also show that the MaxSAT formulation is well adapted for boosting using the well-known AdaBoost Algorithm.

1 Introduction

Decision tree is still a very popular machine learning model for a number of reasons. A first important element in favour of decision trees is their interpretability, which, in some settings might be essential even at the cost of a lower accuracy. The wide range of efficient methods to compute decision trees is also a decisive factor to their success. Moreover, models using decision trees within ensemble methods, such as (deep) random forests, can have state of the art accuracy on a number of benchmarks [Zhou and Feng, 2017].

The most widely used techniques for computing decision trees (e.g. `CART` [Breiman *et al.*, 1984], `ID3` [Quinlan, 1986] and `C4.5` [Quinlan, 1993]) are greedy heuristics, typically build trees from top to bottom selecting the feature yielding the highest information gain. However, it was observed early that significantly simpler trees than those computed by greedy methods might exist [Bennett, 1994]. Moreover, simpler (e.g. smaller) trees, are often more accurate on unknown data.

Recently, several exact methods have been introduced to find optimal decision trees for some combinations of criteria involving, their size (i.e., nodes), depth, and their empirical accuracy. However, despite the advantages of optimal trees, these exact algorithms have not yet replaced greedy heuristics as the most popular method. Firstly, even though great advances have been made on exact algorithms, scalability is still an issue. Moreover, one must use proxy criteria which may be more or less relevant in practice. For instance, some approaches (e.g., [Bessiere *et al.*, 2009; Narodytska *et al.*, 2018; Avellaneda, 2020]) minimize some size criterion, either number of nodes or maximum depth, subject to the constraint that the tree is perfectly accurate on the training set. However, this approach is often criticized as it requires consistent data sets, may entail overfitting and produces unnecessarily large trees. Indeed, greedy methods often use pruning techniques which compromise the accuracy on the training set for (often) better generalization performance. Therefore, other methods [Nijssen and Fromont, 2007; Bertsimas and Dunn, 2017; Verwer and Zhang, 2019; Verhaeghe *et al.*, 2019; Aglin *et al.*, 2020; Hu *et al.*, 2019] opt instead for optimizing accuracy (on the training set) subject to constraints on the maximum depth. This approach seems better suited to most usages of decision trees since it deals naturally with noisy data sets, and the trees produced are smaller and usually more accurate. However, restricting the constraints on the tree structure to a maximum depth is very restrictive and usually entails that if the "best" tree is narrow and deep, finding it will be very costly. For instance, the SAT-encoding in [Avellaneda, 2020] can be easily adapted to use number of meaningful nodes as objective, however, the encoding will grow exponentially in the depth and therefore intractable.

In this paper, we show that a simple adaptation of the SAT model introduced by Narodytska *et al.* to a MaxSAT formula, offsets most of its drawbacks and makes it a very attractive method in many cases. Firstly, it makes it easy to use more practical objective functions based on accuracy. In fact, this formulation allows to use any linear combination of size, depth and accuracy as objective. Moreover, and somewhat counter-intuitively, it goes some way toward solving the scalability issue. Solving the MaxSAT formula to optimality is of course harder than solving the corresponding SAT formula.

*Corresponding author

However, dropping the constraint of perfect accuracy means that the tree, and hence the formula, can be smaller. In addition, when using state-of-the-art MaxSAT solvers [Berg *et al.*, 2019], we observe that we can handle much larger formulas. This model finds smaller trees than the standard greedy methods in reasonable time on data sets that are out of reach for the previous SAT method from [Narodytska *et al.*, 2018]. Finally, the MaxSAT approach can be integrated within a classical boosting method easily to improve the prediction performance. This technique still improves the scalability because individual trees of the ensemble can be smaller. Our experimental evaluation outlines the overfitting phenomenon due to the requirement of perfect accuracy in earlier approaches, and shows that the proposed approach is competitive with the state-of-the-art heuristics and exact methods.

The rest of the paper is organised as follows. In Section 2, we give the technical background of the paper. Then, we present our MaxSAT propositions in Section 3. Finally, in Section 4, we show details of our experimental results .

2 Technical Background

We present in this section a brief formal background to introduce the different notations used throughout the paper.

2.1 Classification via Decision Trees

Using for instance the fairly standard one-hot-encoding and other standard techniques, non-binary categorical features and numerical features of a dataset can be transformed into binary values. In this paper, we then focus on binary classification for data with binary features and we use similar notations to those of [Narodytska *et al.*, 2018].

Let $\mathcal{F} = \{f_1, \ldots, f_K\}$ be a set of K binary features, all of them take value in $\{0, 1\}$. The training data, denoted by $\mathcal{E} = \{e_1, \ldots, e_M\}$, is a set of M examples. The examples in $\mathcal{E}$ are partitioned into $\mathcal{E}^+$ and $\mathcal{E}^-$, denoting respectively positives examples and negative examples. Precisely, an example $e_q \in \mathcal{E}$ is represented as a 2-tuple (x_q, c_q), where $x_q \in \{0,1\}^K$ denotes the value vector for all binary features associated with the example and $c_q \in \{0, 1\}$ is the class of the example. We have $c_q = 1$ if $e_q \in \mathcal{E}^+$ and $c_q = 0$ if $e_q \in \mathcal{E}^-$.

The classification problem is to learn some function h (the *classifier*) mapping value vectors to class values, matching as accurately as possible the actual function ϕ on the training data (i.e., $\phi(x_q) = c_q$) and generalizes well to unseen test data. We shall use the term *learner* in Section 2.2 to denote the method to learn the classifier. In this paper, the function h we learn is represented by a binary and full *decision tree* (DT). That is, a binary tree where every internal node has exactly two children. Every internal node is associated to a feature or its negation and every leaf node is associated to a class.

2.2 Ensemble Methods

Ensemble methods is a set of methods that train multiple learners and then combine them for getting better predictions than a single classifier [Zhou, 2012]. We distinguish two families of ensemble methods: (1) Boosting methods where the learners draw from the same data set and are dependent on each other: (2) Bagging methods where the learners draw from different independent samplings and build classifiers independently from each other.

As a typical Boosting method, AdaBoost [Freund and Schapire, 1997] constructs T learners in a sequence of T iterations. Every iteration t learns a classifier h_t. In each iteration t, the data distribution $\mathcal{D}_t$ indicating the importance of each example is updated using a coefficient $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$ where ϵ_t is the error rate of the tth iteration.

Let ϕ be the actual function, and Z_t a normalization factor. AdaBoost sets the initial distribution $\mathcal{D}_1(x_q) = 1/|\mathcal{E}|$ for each example $e_q = (x_q, c_q) \in \mathcal{E}$. Then it updates the data distribution for the $(t+1)$-th iteration $\mathcal{D}_{t+1}$ based on the t-th data distribution $\mathcal{D}_t$ using the equation (1):

$$\mathcal{D}_{t+1}(x_q) = \frac{\mathcal{D}_t(x_q)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(x_q) = \phi(x_q) \\ \exp(\alpha_t) & \text{if } h_t(x_q) \neq \phi(x_q) \end{cases} \quad (1)$$

Intuitively, examples that were misclassified in previous iterations will receive more attention in future iterations than those correctly classified. The final predictions are based on weighted voting where every learner h_t is associated to a weight α_t. That is, if H denotes the aggregated prediction, then H is calculated using equation (2).

$$H(x_q) = 1 \text{ if } \sum_{t=1}^{T} \alpha_t g(h_t(x_q)) > 0 \text{ and } 0 \text{ otherwise} \quad (2)$$

where $g(0) = -1$, $g(1) = 1$.

In reverse, Bagging [Breiman, 1996] constructs independent learners using different subsets (with the same size) from the original dataset. The data sampling is performed with replacement and the final prediction uses majority voting.

2.3 SAT & MaxSAT

We give a brief background on (Maximum) Boolean Satisfiablity and we refer the reader to [Biere *et al.*, 2009] for a comprehensive reference. Boolean Satisfiablity (SAT) is a declarative approach to solve decision problems expressed with Boolean variables and clauses. The variables capture the unknowns of the problem and the clauses represent the constraints to satisfy. A *Boolean variable* var takes a value in a domain $Dom(var) = \{0, 1\}$ (or equivalently in $\{false, true\}$). A *literal* is a Boolean variable or its negation. A *clause* is a disjunction of literals. A value $val \in \{0, 1\}$ is *assigned* to a variable var if $Dom(var) = \{val\}$. An assignment satisfies a clause if one of its literals is true. Given a set of variables $\mathcal{X}$ and a set of clauses defined over $\mathcal{X}$, the SAT problem can be defined as finding an assignment of the variables such that all the clauses are satisfied.

Maximum Satisfiability (MaxSAT) is an optimization version of SAT. The MaxSAT problem is to find an assignment of the variables maximizing the number of satisfied clauses. Here we consider the weighted partial MaxSAT problem, where clauses are separated into *hard* clauses (clauses that must be satisfied) and *soft* clauses (clauses that can be violated). Every soft clause is associated to a weight (a positive integer). In weighted Partial MaxSAT, the problem is to find an assignment that satisfies all hard clauses, and maximizes the sum of weights of soft clauses that are satisfied.

2.4 SAT-based Learning of Decision Trees

Consider the following decision problem:

$P(\mathcal{E}, N)$ *: Given a set of examples* $\mathcal{E}$*, is there a full binary decision tree of size* N *that classifies correctly every example in* $\mathcal{E}$*?*

In [Narodytska *et al.*, 2018], a SAT model for the problem $P(\mathcal{E}, N)$ is proposed. This model can be used as an oracle to compute a decision tree of minimal size that classifies every example. The encoding consists of three parts:

- **Part 1:** Constraints on the structure of the tree
- **Part 2:** Constraints for mapping features (respectively, classes) to internal nodes (respectively, leaf nodes)
- **Part 3:** Constraints for correctly classifying every example in the training set

We refer the reader to the original paper for the full model. In the following, we describe a part of the model needed to explain the changes proposed in Section 3.1. Let N be the size of the tree. For all $i = 1, \ldots, N$, $LR(i)$ denotes the set of even integers in $[i+1, min(2i, N-1)]$ and $RR(i)$ denotes the set of odd integers in $[i+2, min(2i+1, N)]$. We consider the following Boolean variables (where $i \in \{1, \ldots, N\}$):

- v_i is $true$ iff node i is a leaf node
- l_{ij} (respectively r_{ij}) is $true$ iff node j is the left (respectively right) child of node i where $i \in \{1, \ldots, N\}$ and $j \in LR(i)$ (respectively $j \in RR(i)$).
- c_i is $true$ iff the class associated to leaf node i is 1
- d^v_{ri} is $true$ iff f_r is discriminated for value v by node i, or by its ancestors, where $r \in [1, K]$, and $v \in \{0, 1\}$ (i.e., the test $[f_r \neq v]$ is used on the branch from the root to node i)

Let $e_q \in \mathcal{E}^+$, and let $\sigma(r, q) \in \{0, 1\}$ be the value of the feature f_r for e_q. The following constraints are used to forbid e_q to be classified by a leaf node i ($i \in [1, \ldots, N]$) associated with a negative class:

$$v_i \wedge \neg c_i \rightarrow \bigvee_{r=1}^{K} d^{\sigma(r,q)}_{r,i} \tag{3}$$

The same idea is used for every example $e_q \in \mathcal{E}^-$:

$$v_i \wedge c_i \rightarrow \bigvee_{r=1}^{K} d^{\sigma(r,q)}_{r,i} \tag{4}$$

3 MaxSAT-based Learning for Decision Trees

By classifying every example in the training set, the previous SAT model naturally introduces overfitting. This issue can be avoided by considering the optimisation problem:

$P^*(\mathcal{E}, N)$: *Given a set of examples* $\mathcal{E}$*, find a full binary decision tree of size* N *that maximises the number of examples in* $\mathcal{E}$ *that are correctly classified.*

In the following we show how the problem $P^*(\mathcal{E}, N)$ can be effectively modelled using MaxSAT. Next, we show adaptations of the model to take care of the maximum/exact depth of the tree and to model a relaxation of the problem where N is an upper bound on the number of nodes.

3.1 MaxSAT Model to Learn Decision Trees

We propose in this section a MaxSAT encoding for the optimisation problem $P^*(\mathcal{E}, N)$. That is, given a set of examples $\mathcal{E}$, find a full binary decision tree of size N that maximises the number of examples in $\mathcal{E}$ that are correctly classified.

All the constraints from part 1 and part 2 of the SAT model discussed in Section 2.4 are kept as hard clauses. Then, for classifying the examples, we introduce one Boolean variable b_q for every example $e_q \in \mathcal{E}$ that is true only if e_q is correctly classified by the tree. Linking the b_q variables with part 3 of the SAT model is done with the following hard clauses:

$$\forall e_q \in \mathcal{E}^+, \forall i \in [1, N], b_q \rightarrow (v_i \wedge \neg c_i \rightarrow \bigvee_{r=1}^{K} d^{\sigma(r,q)}_{r,i}) \tag{5}$$

$$\forall e_q \in \mathcal{E}^-, \forall i \in [1, N], b_q \rightarrow (v_i \wedge c_i \rightarrow \bigvee_{r=1}^{K} d^{\sigma(r,q)}_{r,i}) \tag{6}$$

Finally, in order to model the objective of maximizing correctly classified examples, each literal b_q is declared as a soft clause. Clearly, the number of satisfied soft clauses is equal to the number of correctly classified examples.

3.2 Constraints for Controlling Depth

A same tree size might clearly lead to different tree topologies. From Figure 1, we can see two extreme situations using the same size 7: a fully unbalanced binary tree (left), and a complete (balanced) binary tree (right). Having a way to control the topology of the tree, via both the depth and the size, is a desired property to have, when learning decision trees. We show how to control the depth of the tree in this section. Note that we count the depth of binary tree from root as depth 0.

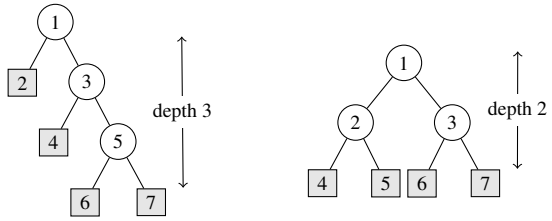


Figure 1: Two Binary Trees of the Same Size with Different Depths

We introduce a Boolean variable $depth_{i,t}$ that is true iff the node i is in depth t. Observe that t can only be in $[\lceil \log(i+1) \rceil - 1, \lceil (i-1)/2 \rceil]$. We present the different constraints to encode the maximum allowed depth of the tree. Notice first that $depth_{1,0}$ has to be true since the root node is always in depth 0. Then every node should be in one exact depth.

$$\sum_{t \in [\lceil \log(i+1) \rceil - 1, \lceil (i-1)/2 \rceil]} depth_{i,t} = 1, \quad \text{for } i = 1, \ldots, N \tag{7}$$

If node i is in depth t, and node j is a child of node i, then node j must be in depth $t+1$.

$$\begin{aligned} &depth_{i,t} \wedge l_{ij} \rightarrow depth_{j,t+1}, \quad \text{for } i = 1, \ldots, N, j \in LR(i) \\ &depth_{i,t} \wedge r_{ij} \rightarrow depth_{j,t+1}, \quad \text{for } i = 1, \ldots, N, j \in RR(i) \end{aligned} \tag{8}$$

If the maximum depth given is U, then all possible nodes in depth U must be leaf nodes.

$$depth_{i,U} \rightarrow v_i, \quad \text{for } i \in [2U, \min(2^{U+1}-1, N)] \tag{9}$$

The following constraint can be added if U is given as an exact depth instead of an upper bound.

$$\bigvee_{i=2U}^{\min(2^{U+1}-1,N)} depth_{i,U} = 1 \tag{10}$$

3.3 Constraints for Controlling Tree Size

We show that the previous encoding can be easily adapted to build a tree with an upper bound on the size of the tree instead of the exact size. Consider firstly the relationship between the size and the depth in a full binary tree. For a maximum depth U, the upper bound of the size is $2^{U+1}-1$. If U is set as exact depth, we additionally get the lower bound for the size as $2U+1$. Observe also that the size of a full and binary decision tree can only be an odd number starting from 3.

Suppose that N is an upper bound on the number of nodes. We introduce a set of Boolean variables m_i (where $i \in \{3, 5, 7 \dots N\}$) to indicate if at least i nodes are used to construct the tree. Recall that we need at least 3 nodes. We therefore need to enforce m_3 to be true. If at least $i+2$ nodes are used then at least i nodes are used:

$$m_{i+2} \rightarrow m_i, \quad \text{for all } i \in [1, N-2] \tag{11}$$

We then apply the following simple rule to adapt each constraint (parts 1 & 2, the new constraints of part 3, and the depth constraints): We look at each clause C from the encoding separately, and consider j, the largest node index used in C. We simply replace the clause C by:

$$\begin{aligned} m_j &\rightarrow C \quad \text{if } j \text{ is odd} \\ m_{j+1} &\rightarrow C \quad \text{if } j \text{ is even} \end{aligned} \tag{12}$$

That is, if j is odd (respectively even) and at least j (respectively $j+1$) nodes are used, then the clause C must hold.

3.4 Integration in AdaBoost

In this section, we explain how the framework of MaxSAT is well adapted to implement AdaBoost. The MaxSAT model used to learn the decision tree at iteration t is identical to the previous model (in which one can control the size and the depth) except for the weight associated to each soft clause b_q. Indeed, we use weighted partial MaxSAT to approximate the data distribution $\mathcal{D}_t$ by associating every soft clause b_q to a positive integer weight w_q^t. The weighted voting follows the original AdaBoost algorithm in Equation (2). We set all weights at the first iteration with the value 1 as initial distribution indicating the equal importance of the examples. Then, the value of w_q^{t+1} is calculated based on w_q^t in two steps as follows. Firstly, we update and normalize the weights:

$$\hat{w}_q^{t+1} = \frac{w_q^t * factor_q^t}{\sum_{q=1}^{M}(w_q^t * factor_q^t)} \tag{13}$$

where $factor_q^t$ is the updating factor based on prediction:

$$factor_q^t = \begin{cases} \exp(-\alpha_t) & \text{if } h_t(x_q) = \phi(x_q) \\ \exp(\alpha_t) & \text{if } h_t(x_q) \neq \phi(x_q) \end{cases} \tag{14}$$

Secondly, we discretize the weight $\hat{w}_q^{t+1}$ as follows:

$$w_q^{t+1} = round(\frac{\hat{w}_q^{t+1}}{\min_{i \in \{1,\dots,M\}}(\hat{w}_i^{t+1}))}) \tag{15}$$

4 Experimental Results

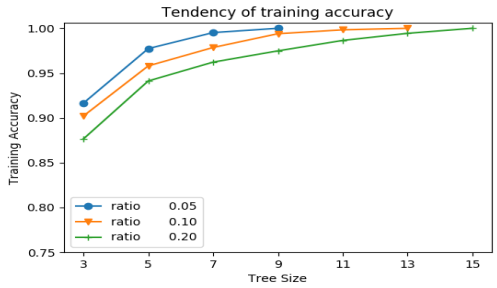
We perform three experiments on datasets from CP4IM[1]. The first experiment aims at highlighting the overfitting behaviour of the SAT approach. In the second experiment we compare the performance with the state of the art heuristic (`CART`) and exact (`DL8.5`) methods. Finally, we evaluate the MaxSAT implementation of Boosting and Bagging. The datasets are binarized with the one-hot-encoding. We ran all experiments using Xeon E5-2695 v3 @ 2.30GHz CPU and running xUbuntu 16.04.6 LTS. Our code source and datasets are available online at https://gepgitlab.laas.fr/hhu/maxsat-decision-trees.

4.1 The Overfitting Phenomenon

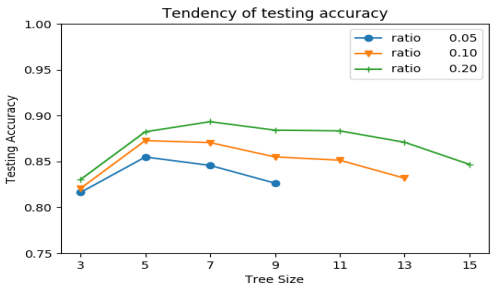
The first set of experiments aims to show the existence of the overfitting behaviour for the SAT method of learning optimal size decision trees that classify every example in the training set. For this, we learn decision trees with our MaxSAT encoding by increasing the size of the trees starting from 3 until we find the size that classifies correctly every example in the training set. The final decision tree obtained with the proposed MaxSAT model corresponds to the SAT model of [Narodytska *et al.*, 2018]. We use the hold-out method to split training set and testing set for all datasets used. Following [Narodytska *et al.*, 2018], we choose (only in this experiment) three (small) different splitting ratios $r = \{0.05, 0.1, 0.2\}$ to generate training sets. And remaining examples are used for testing. This process is repeated 10 times using different randomisation seeds. We use the RC2 MaxSAT solver [Ignatiev *et al.*, 2019]. In the context of these experiments, the solver is left with no limit until it finds the optimal solution (in terms of training accuracy).

In Figure 2, we report the average training accuracy and testing accuracy for one dataset using the different sampling ratios. We can see that the training accuracy increases with the tree size, until reaching a perfect classification. Notice that the tree that classifies every example has the same size as the one using the original SAT approach of [Narodytska *et al.*, 2018]. The testing accuracy shown in Figure2(b) shows that the perfect tree overfits the training data, since smaller trees, while less accurate on the training set have a better testing accuracy. This phenomenon is not as marked for every dataset, however, we almost systematically observe a plateau whereby the testing accuracy stays constant at best while the training accuracy increases.

[1] https://dtai.cs.kuleuven.be/CP4IM/datasets/



(a) Training accuracy as a function of the tree size



(b) Testing accuracy as a function of the tree size

Figure 2: Tendency of training accuracy (a) and testing accuracy (b) for the "breast-cancer" in different ratios

4.2 Comparison with Different Methods

In this section, we compare the prediction accuracy of the decision trees learnt by MaxSAT with `CART` [Breiman *et al.*, 1984] as a state-of-the-art heuristic method (from the scikit-learn python library [Pedregosa *et al.*, 2011]) and `DL8.5` [Aglin *et al.*, 2020] as a state-of-the-art exact approach via its python package (version 0.0.9). Their parameters are kept to their default values, except for the maximum depth which is a fixed parameter for all the models. We use stratified sampling to preserve the class distribution with 5-fold cross-validation. This process is repeated with 10 different randomisation seeds for each dataset. We use the MaxSAT solver Loandra [Berg *et al.*, 2019; Demirović and Stuckey, 2019] as RC2 did not scale as well on the datasets we used. Indeed RC2 is a bottom-up approach, whereas Loandra can return good solutions within a limited time. The timeout for training is set to 15 minutes and the memory limit to 16GB for all the experiments in this section. The core guided phase in Loandra is limited to 10 minutes. We choose three different maximum depths $\mathbf{d} \in \{2, 3, 4\}$ for **`CART`** and **`DL8.5`**. We test two version of the MaxSAT model: **DT(exact)** the model that learns trees with an exact depth (in the set $\{2, 3, 4\}$); and **DT(max)** the model that learns trees with an upper bound on the depth.

The accuracy of the different methods are reported in Table 1. The second column shows the size of dataset (#s) and the number of binary features (#f_b). The third column shows the chosen maximum depth. The column "Acc" stands for accuracy in percent, "Opt" stands for the percentage of optimality, and "Time" stands for the run time in seconds for instances proven to optimality. The value "MO" indicates a memory out. The value "TO" indicates a time out. The best values among all methods are in bold. Testing and training accuracies of DT(max) and DT(exact) are marked with * if they are within 3 percentage points of the best.

We do not report exact run time of `CART` cause it takes only few seconds. In terms of training and testing accuracy, Table 1 shows that the trees learnt via MaxSAT encoding are competitive with both heuristic and exact methods in prediction performance. Although MaxSAT model can not always report optimality, it is close to the optimal solution obtained by `DL8.5`. In addition, `DL8.5` needs massive memory for deep trees, while the MaxSAT model does not. For instance, for a maximum depth of 5, `DL8.5` runs out of memory on 6 datasets, even when lifting the limit to 50GB.

4.3 Boosting via MaxSAT

In order to show the impact of our AdaBoost implementation with MaxSAT, we compare it to bagging, a single learner using MaxSAT, and AdaBoost based on `CART`. We fix the parameters (size, depth, timeout) for every learner to be the same. In particular, we use the MaxSAT model with an upper bound for the depth (i.e., DT(max)) using the Loandra solver and a timeout of 15 minutes. The number of learners (for boosting and bagging) is set yo be 21 as reasonable. We chose datasets where the MaxSAT models do not perform well from the results in Table 1. We use the hold-out method with ratio $r = 0.8$ to split each dataset into training and testing set, which is repeated with 10 different randomisation seeds.

The results are shown in Table 2 (where BS stands for the AdaBoost model via MaxSAT, BSH stands for the AdaBoost model based on `CART`, and BG stands for the Bagging model). The accuracy values are shown in percent. The best values between the three methods are in bold. This table clearly shows that the Boosting via MaxSAT does improve the training accuracy in almost all the cases with respect both to bagging and to a single learner. The boosted trees via MaxSAT generalises very well and are competitive with those based on `CART`.

5 Conclusion

We propose an exact method for learning decision trees based on the MaxSAT framework. Our initial motivation behind this work was to avoid the overfitting issue with the previous SAT encoding and to address the scalability issue. The declarative nature of MaxSAT made it possible to additionally control the size and the depth of the learnt trees. The empirical results show that our approach learns good quality decision trees compared to heuristics method and the recent state-of-the-art. We also show that our MaxSAT-based method is well suited for applying boosting techniques.

In the future, it would be interesting to guide the branching heuristic using the information gain coefficient. Furthermore, we will explore the integration of our method into the Random Forest framework [Breiman, 2001].

Acknowledgements

The authors would like to thank Nina Narodytska for kindly sharing the source code of the original SAT model.

Datasets	$\#s$ / $\#f_b$	d	Testing Accuracy				Training Accuracy								
			DT(exact)	DT(max)	CART	DL8.5	DT(exact)			DT(max)			CART	DL8.5	
			Acc	Acc	Acc	Acc	Acc	Opt	Time	Acc	Opt	Time	Acc	Acc	Time
anneal	812 / 89	2	82.16*	82.05*	81.09	**82.31**	**83.19**	86	593.71	**83.19**	93.3	530.45	81.48	**83.19**	0.03
		3	84.23*	84.29*	81.22	**85.26**	85.06*	0	TO	85.05*	0	TO	81.60	**86.26**	1.76
		4	83.71*	84.50*	80.94	**86.42**	86.11*	0	TO	85.79	0	TO	82.68	**89.10**	73.53
audilogy	216 / 146	2	94.82*	94.49*	94.56	**94.92**	**95.44**	**100**	25.84	**95.44**	**100**	25.17	94.91	**95.44**	0.04
		3	93.72*	93.91*	**94.16**	93.53	97.87*	0	TO	97.84*	0	TO	97.33	**98.07**	2.42
		4	**94.70**	94.08*	94.51	94.50	94.43	36	706.00	99.43*	34	777.25	98.89	**99.90**	46.33
australian-credit	653 / 124	2	84.72*	84.89*	**86.59**	84.81	87.00*	12	866.79	87.00*	12	883.78	86.68	**87.01**	0.06
		3	84.92*	85.31*	84.99	**85.33**	87.83*	0	TO	87.79*	0	TO	86.88	**89.00**	6.96
		4	85.00*	85.31*	**85.44**	MO	88.44*	0	TO	88.19*	0	TO	**89.04**	MO	MO
breast-cancer	683 / 89	2	**93.97**	93.92*	93.89	93.88	**94.94**	**100**	2.90	**94.94**	**100**	3.19	94.42	**94.94**	0.02
		3	94.07*	**94.45**	93.80	94.14	96.66*	20	842.64	96.66*	7	844.62	95.66	**96.67**	0.71
		4	**94.05**	93.95*	93.79	93.66	97.70*	0	TO	97.39*	0	TO	96.91	**98.10**	20.38
car	1728 / 21	2	**85.53**	**85.53**	**85.53**	**85.53**	**85.53**	**100**	3.48	**85.53**	**100**	3.95	**85.53**	**85.53**	0.01
		3	87.49*	87.49*	**87.65**	87.49	**89.24**	96	591.65	**89.24**	74	700.19	88.48	**89.24**	0.03
		4	89.69*	89.84*	87.93	**90.69**	91.60*	0	TO	91.47*	0	TO	89.62	**92.31**	0.31
heart-cleveland	296 / 95	2	71.59*	71.60*	**72.71**	71.53	80.94*	0	TO	**80.95**	0	TO	78.20	**80.95**	0.03
		3	76.08	76.09	**79.37**	75.85	83.95	0	TO	85.00*	0	TO	85.78	**87.20**	2.99
		4	75.11	75.82*	76.55	**78.15**	85.57	0	TO	85.31	0	TO	88.32	**92.77**	134.72
hypothyrold	3247 / 86	2	**97.84**	**97.84**	**97.84**	**97.84**	**97.84**	**100**	130.88	**97.84**	**100**	131.56	**97.84**	**97.84**	0.04
		3	97.83*	97.83*	**97.87**	97.86	98.13*	0	TO	98.13*	0	TO	98.12	**98.14**	2.95
		4	97.98*	98.04*	**98.10**	97.87	98.36*	0	TO	98.35*	0	TO	98.38	**98.44**	121.92
kr-vs-kp	3196 / 73	2	**86.92**	**86.92**	76.75	**86.92**	**86.92**	**100**	130.88	**86.92**	**100**	131.56	77.49	**86.92**	0.03
		3	93.57*	93.56*	90.43	**93.75**	93.60*	0	TO	93.61*	0	TO	90.43	**93.81**	1.60
		4	93.69*	94.10*	94.09	**95.36**	93.80*	0	TO	94.20*	0	TO	94.09	**95.50**	67.18
lymph	148 / 68	2	79.16*	79.42*	**80.85**	79.07	**86.07**	**100**	31.32	**86.07**	**100**	35.98	84.58	**86.07**	0.01
		3	80.95*	80.45*	79.05	**81.80**	91.71*	0	TO	91.78*	0	TO	89.97	**92.81**	0.44
		4	80.53*	81.71*	**82.15**	80.28	94.51*	0	TO	94.87*	2	868.11	94.88	**99.00**	7.75
mushroom	8124 / 112	2	**96.90**	**96.90**	92.71	**96.90**	**96.90**	**100**	89.89	**96.90**	**100**	132.94	92.71	**96.90**	0.09
		3	**99.97**	99.66*	96.53	99.90	99.73*	0	TO	99.69*	0	TO	96.55	**99.90**	4.94
		4	**100**	99.98*	99.90	**100**	**100**	**100**	354.33	99.99*	96	388.80	99.92	**100**	28.65
primary-tumor	336 / 31	2	79.55*	79.82*	**80.15**	80.06	**83.07**	**100**	17.16	**83.07**	**100**	17.74	82.82	**83.07**	0.01
		3	82.79*	82.61*	79.83	**83.27**	86.32*	0	TO	86.44*	0	TO	84.81	**86.58**	0.10
		4	82.74*	83.24*	81.72	**83.30**	87.45*	0	TO	87.71*	0	TO	87.45	**90.22**	1.57
soybean	630 / 50	2	**91.27**	**91.27**	87.94	**91.27**	**91.27**	**100**	8.54	**91.27**	**100**	8.56	89.29	**91.27**	0.01
		3	94.29*	94.19*	90.46	**94.35**	95.46*	0	TO	95.50*	0	TO	92.23	**95.51**	0.20
		4	95.30*	96.04*	92.46	**96.17**	97.14*	0	TO	97.05*	0	TO	94.39	**98.02**	3.58
splice-1	3190 / 287	2	83.44*	83.22*	**84.04**	82.80	84.13*	0	TO	84.17*	0	TO	84.04	**84.31**	0.50
		3	83.62	85.60	90.96	**92.83**	84.06	0	TO	85.90	0	TO	91.31	**92.98**	81.01
		4	85.87	83.18	**95.25**	MO	86.21	0	TO	83.91	0	TO	**95.43**	MO	MO
tic-tac-toe	958 / 27	2	67.45*	67.51*	**68.22**	67.59	**71.12**	**100**	99.50	**71.12**	**100**	92.00	70.92	**71.12**	0.01
		3	**73.54**	73.42*	72.04	72.22	77.45*	0	TO	76.94*	0	TO	75.70	**78.65**	0.08
		4	78.15*	78.04*	**80.97**	80.30	82.05	0	TO	81.33	0	TO	83.82	**86.65**	1.14
vote	435 / 48	2	94.94*	94.98*	**95.44**	94.84	**96.22**	**100**	3.49	**96.22**	**100**	3.61	95.63	**96.22**	0.01
		3	94.16*	94.18*	**94.67**	93.95	97.39*	0	TO	97.35*	0	TO	96.91	**97.50**	0.23
		4	**94.64**	94.25*	94.57	93.61	98.50*	0	TO	98.51*	0	TO	98.03	**99.18**	4.04

Table 1: Evaluation of the MaxSAT Model with `CART` and `DL8.5`

Datasets	d	Testing accuracy				Training accuracy			
		DT(max)	BS	BSH	BG	DT(max)	BS	BSH	BG
anneal	2	80.98	80.98	**83.99**	80.98	83.20	83.36*	**85.21**	83.17
	3	82.82	84.90*	**86.81**	82.82	84.90	**88.60**	88.26	85.00
	4	82.82	84.82*	**87.55**	83.30	85.52	90.14*	**91.05**	85.81
australian-credit	2	87.12	**87.20**	85.98	89.20	86.56	89.98	**93.84**	85.89
	3	87.88	87.54*	87.05	**88.72**	87.33	89.66	**93.80**	87.03
	4	87.88	87.88*	85.91	**89.14**	88.48	90.21	**99.31**	87.03
car	2	84.68	**96.53**	95.29	84.68	85.75	**97.47**	96.27	85.75
	3	87.86	95.44*	**97.83**	89.92	89.15	97.10*	**98.84**	91.08
	4	90.46	98.36*	**98.67**	95.02	91.24	98.70*	**99.91**	95.30
heart-cleveland	2	73.33	78.33*	**79.83**	76.85	80.93	**90.25**	90.13	85.22
	3	78.33	**83.70**	81.17	81.48	83.90	93.97	**99.15**	88.37
	4	**85.0**	80.17	77.83	83.89	83.47	96.09	**100**	90.07
primary-tumor	2	**82.35**	**82.35**	81.91	82.03	82.84	86.19*	**87.05**	83.25
	3	82.65	**84.48**	80.74	84.31	86.34	**90.92**	89.96	87.77
	4	80.88	79.14	78.97	**84.80**	86.57	88.81	**93.77**	89.59
tic-tac-toe	2	68.91	77.14*	**78.08**	71.27	70.98	80.39*	**82.61**	72.37
	3	74.61	**94.69**	93.01	81.46	76.86	95.96*	**96.31**	89.46
	4	76.74	94.49*	**96.94**	85.26	81.31	95.80	**100**	85.37

Table 2: Evaluation of the MaxSAT Boosting Approach

References

[Aglin *et al.*, 2020] Gael Aglin, Siegfried Nijssen, and Pierre Schaus. Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of the Thirty-Fourth Conference on Artificial Intelligence (AAAI)*, New York, USA, February 2020.

[Avellaneda, 2020] Florent Avellaneda. Efficient inference of optimal decision trees. In *Proceedings of the Thirty-Fourth Conference on Artificial Intelligence (AAAI)*, New York, USA, February 2020.

[Bennett, 1994] Kristin Bennett. Global tree optimization: A non-greedy decision tree algorithm. In *Computing Science and Statistics*, pages 156–160, 1994.

[Berg *et al.*, 2019] Jeremias Berg, Emir Demirović, and Peter J. Stuckey. Core-boosted linear search for incomplete maxsat. In *Proceedings of the 16th International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research - CPAIOR*, pages 39–56, June 4-7, 2019.

[Bertsimas and Dunn, 2017] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[Bessiere *et al.*, 2009] Christian Bessiere, Emmanuel Hebrard, and Barry O'Sullivan. Minimising decision tree size as combinatorial optimisation. In *Proceedings of the of 15th International Conference on Principles and Practice of Constraint Programming - CP*, pages 173–187, September 20-24, 2009.

[Biere *et al.*, 2009] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, 2009.

[Breiman *et al.*, 1984] Leo Breiman, J. H. Friedman, R. A. Olshen, , and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, 1st edition, 1984.

[Breiman, 1996] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

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

[Demirović and Stuckey, 2019] Emir Demirović and Peter J. Stuckey. Techniques inspired by local search for incomplete maxsat and the linear algorithm: Varying resolution and solution-guided search. In Thomas Schiex and Simon de Givry, editors, *Principles and Practice of Constraint Programming - 25th International Conference, CP 2019, Stamford, CT, USA, September 30 - October 4, 2019, Proceedings*, volume 11802 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2019.

[Freund and Schapire, 1997] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[Hu *et al.*, 2019] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14*, pages 7265–7273, December 2019.

[Ignatiev *et al.*, 2019] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. Rc2: an efficient maxsat solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11:53–64, 09 2019.

[Narodytska *et al.*, 2018] Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and João Marques-Silva. Learning optimal decision trees with SAT. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence - IJCAI*, pages 1362–1368, July 13-19, 2018.

[Nijssen and Fromont, 2007] Siegfried Nijssen and Elisa Fromont. Mining optimal decision trees from itemset lattices. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD*, page 530–539, August 12-15, 2007.

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

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

[Quinlan, 1993] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[Verhaeghe *et al.*, 2019] Hélène Verhaeghe, Siegfried Nijssen, Gilles Pesant, Claude-Guy Quimper, and Pierre Schaus. Learning optimal decision trees using constraint programming. In *Proceedings of the 25th International Conference on Principles and Practice of Constraint Programming - CP*, Sept 30 - Oct 4, 2019.

[Verwer and Zhang, 2019] Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In *The Thirty-Third National Conference on Artificial Intelligence (AAAI)*, pages 1625–1632, 2019.

[Zhou and Feng, 2017] Zhi-Hua Zhou and Ji Feng. Deep forest: Towards an alternative to deep neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence - IJCAI*, page 3553–3559, August, 19-25, 2017.

[Zhou, 2012] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition, 2012.