IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Learning Explainable Decision Rules via Maximum Satisfiability

HENRIK E. C. CAO[1], RIKU SARLIN[2], AND ALEXANDER JUNG.[3], (Member, IEEE)

[1]Aalto University, Espoo, Finland
[2]Social Insurance Institution of Finland, Helsinki, Finland
[3]Aalto University, Espoo, Finland

Corresponding author: H. Cao (henrik.cao@aalto.fi)

This work was supported in part by the Social Insurance Institution of Finland

ABSTRACT Decision trees are a popular choice for providing explainable machine learning, since they make explicit how different features contribute towards the prediction. We apply tools from constraint satisfaction to learn optimal decision trees in the form of sparse k-CNF (Conjunctive Normal Form) rules. We develop two methods offering different trade-offs between accuracy and computational complexity: one offline method that learns decision trees using the entire training dataset and one online method that learns decision trees over a local subset of the training dataset. This subset is obtained from training examples near a query point. The developed methods are applied on a number of datasets both in an online and an offline setting. We found that our methods learn decision trees which are significantly more accurate than those learned by existing heuristic approaches. However, the global decision tree model tends to be computationally more expensive compared to heuristic approaches. The online method is faster to train and finds smaller decision trees with an accuracy comparable to that of the k-nearest-neighbour method.

INDEX TERMS classification, combinatorial optimization, decision tree, explainable AI, machine learning, weighted MaxSAT

I. INTRODUCTION

EXPLAINABLE artificial intelligence (XAI) is a family of methods focusing on the algorithmic transparency and interpretability of decision procedures [1], [2]. The ability to explain the decision of a machine learning (ML) algorithm is a vital component of diagnostic, feedback, and human-in-the-loop systems [3], [4], [5]. Furthermore, with increasing statutory restrictions planned on the use of ML methods in customer-oriented applications, there has been renewed interest in the field [6], [7].

Transparency in ML algorithms can be achieved in a number of direct and indirect ways [8]. This work pursues methods whose underlying classification rules are constructed from logical clauses of simple operators. The resulting decision tree models comprise an enumeration of feature combinations leading to a positive prediction. At moderate complexity, tree models can provide an intuitive explanation of the prediction.

Decision trees are aggregates of classification paths propagating through test nodes or decision nodes. A node represents a binary test (or "yes/no question") assessing the state of a single feature, e.g. "Is your monthly income less than 2400€?". By modelling these nodes as propositional variables, we can represent a decision tree in propositional logic as a conjunction of disjunctions of variables. Also known as k-cnf formulae, such formulae can provide a compact and interpretable representation of classification rules over a set of features.

A recent line of work, adapting ideas from Boolean compressed sensing, focuses on learning sparse k-cnf rules via linear programming, [9], [10], [11]. Following advancements in SAT-solving, the problem has also been cast into various maximum-satisfiability (MAX-SAT)-based frameworks [12], [13], [14]. Other works consider learning *decision lists* and *decision sets*, which are generalizations of k-cnf rules [15], [16].

This work extends the MAX-SAT-based framework for learning k-cnf rules proposed in [13]. In particular, we generalize their framework to non-constant clause weights and add cardinality constraints for more efficient recovery. While this allows for finding solutions over much larger datasets, the global model still suffers from the inherent NP-hardness of SAT solving. Therefore, we propose a locally weighted learning approach for query-specific applications [17], [18].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3041040, IEEE Access

Similar to *lazy decision trees* proposed in [19], our local model constructs a k-cnf rule within the neighbourhood of a query data point.

We empirically validate our methods on a dataset of Housing Benefit Applications provided by the Social Insurance Institution of Finland (Kela). The task is to classify benefit applications as accepted/rejected. In addition, an explanation of the decision is provided by the classification rule. We also consider six publicly available datasets.

II. METHODS

A. OPTIMAL DECISION TREES

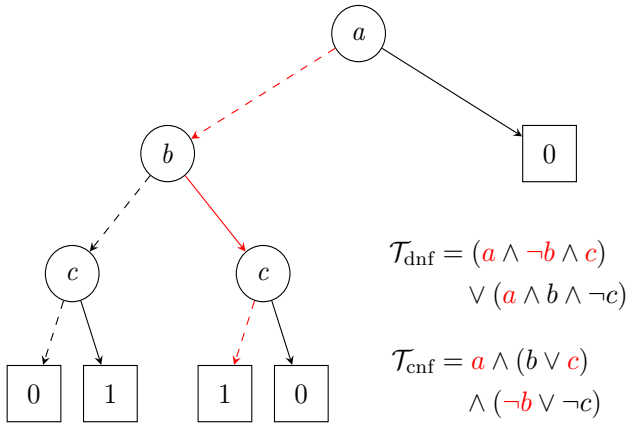

FIGURE 1. A binary decision tree and corresponding propositional formulae in CNF and DNF. True assignments are marked as solid edges, False assignments as dashed. One possible classification path $(a = 1, b = 0, c = 1)$ is marked in red.

Decision trees (Fig.1) are directed acyclic graphs whose internal nodes represent propositional variables (e.g., v = "income < 5000") and whose edges represent assignments to source nodes. We use the term *literal*, e.g., t to denote a propositional variable v or its negation $\overline{v}$. In propositional logic, decision trees can be modelled as formulae in *conjunctive normal form* (CNF) $\mathcal{T}_{\text{cnf}} = \bigwedge T_k$, where clauses $T_k = \bigvee t_k^m$ are disjunctions of literals of variables in the tree and where variables model tests for input features. (Alternatively, a decision tree can be represented in *disjunctive normal form* (DNF) as $\mathcal{T}_{\text{dnf}} = \bigvee T_k$, where $T_k = \bigwedge t_k^m$ is modelled as a conjunction of literals.) We only consider binary variables which can either be true or false.

Let $\mathcal{T} = \{\{t_k^1, \ldots, t_k^{2L}\} : 1 \leq k \leq K\}$ and $\sigma : \mathcal{T} \to \{0, 1\}$ be a *valuation*. We define a decision tree (in CNF) in terms of a valuation σ as

$$\mathcal{T}_\sigma = \bigwedge_k \bigvee_{\sigma(t_k^m)=1} t_k^m \tag{1}$$

Let $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ be a dataset with numeric feature vectors $\mathbf{x}_i \in \mathbb{R}^M$ and labels $y_i \in \{0, 1\}$. Such a feature could be the age of a person or their household income. We quantize a feature x_j using a set of thresholds $\mathbf{a}_j = \{a_j^1, \ldots, a_j^{|\mathbf{a}_j|}\}$ to produce $|\mathbf{a}_j|$ binary features $\boldsymbol{\tau}_{\mathbf{a}_j} = \tau_{a_j^1}, \ldots, \tau_{a_j^{|\mathbf{a}_j|}}$. We will discuss how to choose the thresholds later in section IV. To avoid notational clutter, we use l to index threshold a_l in $\{a_1, \ldots a_L\} = \{a_1^1, \ldots, a_M^{|\mathbf{a}_M|}\}$, where $L = \sum_{j=1}^M |\mathbf{a}_j|$.

We classify a data point x over a decision tree $\mathcal{T}_\sigma$ as

$$\mathcal{T}_\sigma(\mathbf{x}) = \bigwedge_k \left(\bigvee_{\substack{1 \leq l \leq L \\ \sigma(t_k^l)=1}} \tau_{a_l} \vee \bigvee_{\substack{1 \leq l \leq L \\ \sigma(t_k^{L+l})=1}} \overline{\tau}_{a_l} \right) \tag{2}$$

An *optimal* decision tree over $\mathcal{D}$ solves

$$\mathcal{T}_\sigma^* \in \underset{\mathcal{T}_\sigma \in \{0,1\}^{K \times 2L}}{\arg\min} \left\{ L_{\mathcal{D}}(\mathcal{T}_\sigma) + \lambda \|\mathcal{T}_\sigma\|_1 \right\} \tag{3}$$

$$\text{s.t.} \quad |T_k| \geq 1 \qquad 1 \leq k \leq K \tag{4}$$

$$\text{s.t.} \quad |T_k^j| \leq 1 \qquad 1 \leq k \leq K, 1 \leq j \leq M \tag{5}$$

with loss function

$$L_{\mathcal{D}}(\mathcal{T}_\sigma) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} w_i \mathbb{I}[\mathcal{T}_\sigma(\mathbf{x}_i) \neq y_i] \tag{6}$$

The regularization parameter λ trades off prediction accuracy against sparsity of decision tree. Sparsity is measured as the number of true literals in clauses in a tree, $\|\mathcal{T}_\sigma\|_1 = \sum_{T_k \in \mathcal{T}} |T_k|$, where $|T_k|$ measures the number of literals assigned as True in clause T_k. For example, $\mathcal{T}_{cnf}$ in Fig. 1 has $\|\mathcal{T}_{cnf}\|_1 = 1 + 2 + 2$. The sparsity of a decision tree determines its generalization ability. In general, we expect a sparse decision tree to be less prone to overfitting. The constraints (4) require clauses T_k to be non-empty while the constraints (5) exclude implied nodes such as τ_a and τ_b for thresholds $a \neq b$ for feature j (e.g. node 'income < 500' implies node 'income < 700'). The weights w_i in (6) are explained in section III. For now we will assume $w_i = 1$.

B. OPTIMAL DECISION TREES VIA PARTIAL MAX-SAT

Owing to the 0-1 loss function in (6), finding direct solutions of (3)-(5) is known to be NP-hard [20]. We reformulate (3)-(5) as a weighted partial MAX-SAT problem, which will enable the use of specialized solvers.

To this end, let $\eta = \{\eta_1, \ldots, \eta_N\}$ and extend σ to $\sigma : \mathcal{T} \cup \eta \to \{0, 1\}$. Let $\omega : \mathcal{F} \to \mathbb{R} \cup \{\infty\}$ be a mapping associating non-negative weights with clauses in $\mathcal{F}$ (to be defined later in this section). It will be convenient to reformulate (3) by using the equivalence $\min(f) = -\max(-f)$, as

$$\mathcal{T}_\sigma^* \in \underset{\mathcal{T}_\sigma \in \{0,1\}^{K \times 2L}}{\arg\max} \left\{ \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} w_i \mathbb{I}[\mathcal{T}_\sigma(\mathbf{x}_i) = y_i] + \lambda \|\overline{\mathcal{T}_\sigma}\|_1 \right\} \tag{7}$$

where $\overline{\mathcal{T}}$ denotes the negation of variables in $\mathcal{T}$. Let

$$\Omega_i = \bigwedge_k \left(\bigvee_{X_i^j \geq a_l} t_k^l \vee \bigvee_{X_i^j < a_l} t_k^{L+l} \right) \tag{8}$$

which lists, for each rule k, the variables matching the quantized representation of datapoint $\mathbf{x}_i$. We can represent

the weighted indicator function $\mathbb{I}[\mathcal{T}_\sigma(\mathbf{x}_i) = y_i]$ by conjoining the paired constraints

$$N_i = \eta_i \qquad \omega(N_i) = w_i \qquad 1 \leq i \leq N \tag{9}$$

$$D_i = \begin{cases} \eta_i \rightarrow \Omega_i, & \omega(D_i) = \infty \quad \text{if } y_i = 1 \\ \eta_i \rightarrow \neg\Omega_i, & \omega(D_i) = \infty \quad \text{if } y_i = 0 \end{cases} \tag{10}$$

The clauses (9) with weights $w_i \in \mathbb{R}$ are selectors of data points in the training dataset. The clauses (10), in turn, regulate the correctness of the model's prediction over the chosen subset. In particular, with slight abuse of notation, we have the equivalence

$$\mathbb{I}[\mathcal{T}_\sigma(\mathbf{x}_i) = y_i] = \sigma(N_i)\sigma(D_i)$$

Similarly, the cardinality constraint $\lambda\|\overline{\mathcal{T}_\sigma}\|_1$ in (7) is incorporated into the clauses

$$T_k^m = \overline{t_k^m} \qquad \omega(T_k^m) = \lambda \quad \forall t_k^m \tag{11}$$

Lastly, we redefine the cardinality constraints (4) and (5) as propositional clauses

$$E_k = \bigvee_{1 \leq m \leq 2L} t_k^m \qquad \omega(E_k) = \infty \qquad 1 \leq k \leq K \tag{12}$$

$$C_k^j = \{t_k^{a_j^1+1}, \ldots, t_k^{a_j^1+|a_j|}\}1 \qquad \omega(C_k^j) = \infty \qquad 1 \leq j \leq M, 1 \leq k \leq K \tag{13}$$

where $\{r_1, \ldots, r_n\}1$ denotes a 1-out-of-n bound on the number of literals $r \in \{r_1, \ldots, r_n\}$ which can be set to True. For efficient encoding of these cardinality constraints, we make use of the following encoding developed in [21]:

$$\begin{aligned} \{r_1, \ldots r_n\}1 = (\overline{r_1} \vee s_1) & \\ \bigwedge_{1<t<n} (\overline{r_t} \vee s_t) \wedge (\overline{s_{t-1}} \vee s_t) & \\ \bigwedge_{1<t\leq n} (\overline{r_t} \vee s_{t-1}). & \end{aligned} \tag{14}$$

In practice, writing the clauses D_i in CNF for negative examples ($y_i = 0$) is less straightforward. This difficulty can be resolved using Tseytin's transformation. We introduce new variables $s_1, \cdots s_K$ and set $D_i = D_i^0 \wedge \bigwedge_k D_i^k$ where $D_i^0 = \overline{\eta_i} \vee \bigvee_k s_k$ and $D_i^k = \bigwedge_{X_i^m=1}(\overline{s_k} \vee \overline{t_k^m})$.

Letting $\mathcal{F} = \mathcal{F}_s \cup \mathcal{F}_h$, where $\mathcal{F}_s = N \cup T$ and $\mathcal{F}_h = D \cup E \cup C$, we have the equivalence of (3) and its partial weighted MAX-SAT form

$$\sigma^* \in \arg\max_{\sigma \models \mathcal{F}_h} \left\{ \sum_{K \in \mathcal{F}_s} \sigma(K)\omega(K) \right\} \tag{15}$$

Optimal decision trees $\mathcal{T}_{\sigma^*}$ in the form of (1) yield *interpretable* decision tree classifiers $\mathcal{T}_{\sigma^*} : \{0,1\}^M \rightarrow \{0,1\}$ over the subset $\{X_i, y_i\}_{\sigma^*(\eta)} \subset \mathcal{D}_{train}$. Interpretability is endowed by the semantic interpretation of variables $t_k^m \in \mathcal{T}$ which inherit meaning according to their represented features. In this way, the model provides an interpretation of the dichotomy in the dataset, as well as a justification for predictions $\hat{y} = \mathcal{T}_{\sigma^*}(x)$ made on unseen data.

III. LOCAL DECISION TREES

The number of constraints (10) grows as $\mathcal{O}(LN)$ with the number of data points N and binary features L. As a consequence, the problem (9)-(13) is unsolvable in practice for large datasets. This motivates a local learning approach where (15) is solved locally in a query-oriented fashion.

Let $x_q \in \mathbb{R}^M$ be a real-valued query and $\tau_q \in \{0,1\}^{2L}$ its binarisation. We construct $\mathcal{D}_{local}$ in the vicinity of $\boldsymbol{x}_q$ by selecting data points $\{\boldsymbol{x}_i, y_i\}_{\mathcal{N}(\boldsymbol{x}_q)}$ in the neighbourhood $\mathcal{N}(\boldsymbol{x}_q) = \{i | d(\boldsymbol{x}_i, \boldsymbol{x}_q) \leq \delta\}$. Distance information, in the feature space, is incorporated into the weights w_i in (6) using a kernel function $\mathcal{K}(d)$, where d is a metric in the (real-valued) feature space. A popular choice is the Gaussian kernel [22]

$$\mathcal{K}(d) = \begin{cases} exp(-d^2) & \text{if } d < \delta \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

with cut-off parameter δ (although in practice we include a fixed number of nearest neighbours).

We will consider the weighted norm

$$d(\boldsymbol{x}_i, \boldsymbol{x}_q) = \sum_{j=1}^{J} \frac{1}{s_j} |x_i^j - x_q^j|. \tag{17}$$

with sample variances s_j over the training dataset.

Classification of a query point $\boldsymbol{x}_q$ amounts to solving (9)-(13) over $\{\boldsymbol{x}_i, y_i\}_{\mathcal{N}(\boldsymbol{x}_q)}$ with weights $w_i = \mathcal{K}(d(\boldsymbol{x}_i, \boldsymbol{x}_q))$ and classifying $\hat{y}_q = \mathcal{T}_\sigma^*(\boldsymbol{\tau}_q)$.

IV. EXPERIMENTS

We validate the global and local models of the previous section on a number of public datasets (Table 1) from the UCI (University of California Irvine) Machine Learning Repository.[1] As a real-life example, we include a dataset of Housing Benefit Applications (HBA) provided by the Social Insurance Institution of Finland. Our experiments were performed using computer resources within the Aalto University School of Science "Science-IT" project.[2] The models were trained using the UWrMaxSat solver [23] version 1.1.2 with a timeout of 2 hours.

A. HOUSING BENEFIT APPLICATION DATASET

The HBA dataset consists of non-identifying data selected from 6250 housing benefit applications received by the Social Insurance Institution of Finland. A typical application comprises binary- and multiple-choice questions, numerical fields (e.g. age, income), and optional text fields. We concern ourselves with the prediction of whether to accept or reject such an application based on categorical and numerical features.

Specifically, we consider the salient features: income, #residents, #children, max expenses, housing expenses, deductibles, and calculated expenses. Feature binarisation is

[1] https://archive.ics.uci.edu/ml/datasets.php

[2] See https://scicomp.aalto.fi/triton/overview/ for architecture specifications

Dataset	medi	park	iono	liver	pima	bank	HBA
Size	116	195	351	579	768	1372	6250
#Features	9	22	33	10	8	4	7

TABLE 1. Overview of datasets used in the experiments.

Dataset	DT	RIPPER	K=2	K=3	K=4	K=5
medi	93.44	96.11	90.86	94.12	96.34	97.52
	70.41	68.16	71.93	**72.76**	70.09	69.10
parkinsons	95.34	96.27	98.21	99.98	100.0	99.74
	86.23	86.78	87.91	87.11	**89.00**	87.34
ionosphere	91.26	91.61	89.36	91.87	93.97	94.86
	89.53	89.17	87.07	87.95	91.71	**92.17**
indian liver	71.69	70.04	72.49	69.39	68.42	67.33
	70.46	69.10	70.39	66.76	65.74	65.08
pima indian	76.35	75.41	76.23	73.41	72.73	73.65
	74.05	74.63	**74.97**	72.03	71.18	73.65
banknote	99.68	99.21	95.56	97.54	98.91	99.64
	98.11	98.15	94.67	96.44	97.91	**99.04**
HBA	100.0	-	97.52	98.11	98.91	99.91
	96.81	-	95.51	95.97	96.81	**97.34**

TABLE 2. Average training accuracies (above) and test accuracies (below) over 5-fold CV for heuristic decision trees (DT), decision trees trained using RIPPER (RIPPER) and optimal decision tree models for $K \in \{2, 3, 4, 5\}$.

done for a fixed number of thresholds, except for discrete features (#residents, #children) for which all thresholds a_j are included. After experimenting with a number of ways of choosing these thresholds, we settled with choosing thresholds a_j as in-between values observed in the training data. In particular, for each feature j, we ordered the training data in ascending numerical order, took the differences between all ordered consecutive pairs, and chose a fixed number of quantiles from these differences.

B. GLOBAL MODEL

We compared the optimal decision tree model of Section II-B with rule size K against the decision tree algorithms CART [24] and RIPPER [25]. For CART, we trained decision trees of both fixed and maximum tree depth using the *DecisionTreeClassifier* algorithm in the *scikit-learn v0.23.1* Python package. For RIPPER, decision trees were trained using the Java Weka (v3.9.2) Jrip classifier [26]. In our optimal decision tree model, we fixed the regularization parameter $\lambda \in [1.0, 0.1, 0.01]$ and the number of binarisation thresholds per numerical features at $|a_j| = 20$; except for the HBA dataset, where we used $|a_j| = 60$. To provide statistically meaningful measures of accuracy, we report 5-fold cross validation scores averaged over 100 realizations of the training set [27].

Table 2 reports the average training accuracies (above) and test accuracies (below) for each dataset. For the *DecisionTreeClassifier* (DT), we report the highest achieved test score over all fixed-depth and unlimited-depth CART classifiers. We only considered RIPPER after relinquishing our rights to the HBA dataset, wherefore its entry is missing for the HBA dataset.

Increasing the number of thresholds $|a_j|$ during the discretization of a dataset tends to increase the accuracy of the model. This trend is plotted in Fig. 2 (left) for the HBA dataset and fixed parameters $K = 5$ and $\lambda = 10^{-2}$. The regularization parameter λ reflects model sparsity and directly impacts the overfitting/underfitting of the model during training. A typical trend for the HBA dataset is plotted in Fig. 2 (right) for $K = 5$ and $|a_j| = 60$.

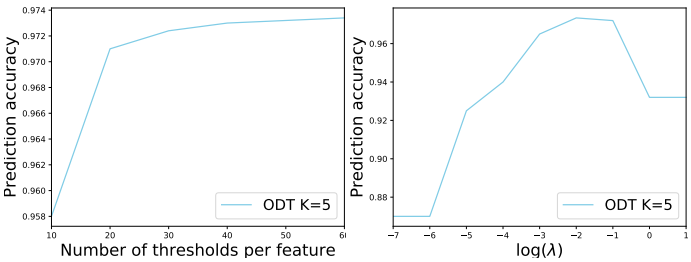

FIGURE 2. Test accuracies of optimal decision tree model for $K = 5$ on the HBA dataset for an increasing number of binary features (left) and regularization parameter (right).

We provide an example 3-clause classification rule extracted by our method on the HBA dataset below.

$$\begin{aligned} &\text{expenses} < 550 \quad \textbf{or} \quad \text{housing exp} < 700 \\ \textbf{and}\ &\text{deductibles} < 1040 \quad \textbf{or} \quad \text{child exp.} < 970 \\ \textbf{and}\ &\text{expenses} < 80 \quad \textbf{or} \quad \text{income} < 1460 \end{aligned} \tag{18}$$

For decision trees larger than $K = 3$, the 2 hour timeout was consistently reached. In comparison, the CART and RIPPER methods trained within milliseconds.

C. LOCAL MODEL

The second experiment considered query-based classification: each query (housing benefit application) was classified on a local dataset of N data points closest to the query. That is, for each query point in the test dataset $\mathcal{D}_{test}$, we subsampled a training dataset $\mathcal{D}_{train}$ of N nearest data points in $\mathcal{D}_{databse}$ based on (17). We then trained the model on $\mathcal{D}_{train}$, as described in Section III, and used it to classify the query point.

Since the model was query-based, we fixed a randomly chosen test set, $|\mathcal{D}_{test}| = 1250$, for each experiment and chose $\mathcal{D}_{databse}$ from the remaining 5000 data points. We again fixed the model parameters $\lambda = 0.1$ and $|a_j| = 10$ and chose $N = 100$ nearest neighbours. Since the training dataset was restricted to a small neighbourhood around the query point, we considered shallow decision trees with $K = 2$ and $K = 3$. The resulting model was compared against the Nearest Neighbourhood algorithm (we used the *scikit-learn 0.23.1* Python package with default settings). We repeated the experiment over 100 realizations of the dataset.

Fig. 3 (top) shows the median classification accuracies and 10th/90th percentiles (shaded) of the models when $|\mathcal{D}_{databse}|$ is increased. Fixing $|\mathcal{D}_{databse}| = 5000$, we also plot the dependencies of our model's classification accuracy on the size of the training dataset (left) and on the number of feature thresholds (right).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3041040, IEEE Access

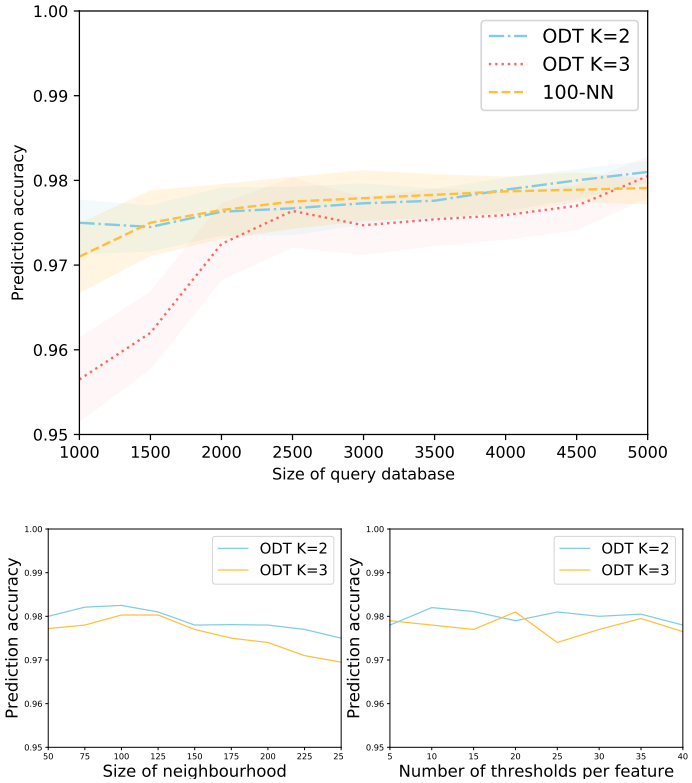

FIGURE 3. Test accuracies for number of neighbours (left), and number of binary features (right).

As expected, increasing the size of the database leads to more accurate models. The local decision tree model performs best on small neighbourhood sizes ($\sim$100) and does not require a large number of thresholds $\mathbf{a}_j$. Average runtimes were around 30 seconds for $K = 2$ and 5 minutes for $K = 3$ respectively.

V. CONCLUSIONS

A. CONCLUSIONS

We have generalized existing methods for constructing weighted optimal decision trees in the form of k-cnf rules via MaxSAT. We also translated this framework into a lazy learning scheme where decision trees are constructed locally in a query-oriented fashion.

Our empirical study (Table 2) supports the belief that optimal methods can be significantly more accurate than heuristic approaches, at the cost of computational efficiency.

Our lazy-learning method (Section III) shows performance comparable to that of the k-nearest-neighbour method (Fig. 3). The method finds optimal decision trees with two clauses ($K = 2$) and small neighbourhood sizes (δ), which allows for particularly fast training.

The explanations (18) are in the form of clauses in a decision tree. It might require additional work to translate these mathematical explanations into proper justifications of decisions on social benefit applications. It is difficult, even impossible, to base required legal reasoning of a benefit decision on the mathematical explanation provided. In spite of this, we believe that these methods could provide sufficient transparency in other applications.

B. FUTURE WORK

A key restriction for learning optimal decision trees is the exponential growth of data constraints. Specializing MAX-SAT solvers to the subclass of problems described by (9)-(13) can make searching more efficient. Furthermore, we expect that exploiting symmetries can help guide the search for candidate solutions. In fact, during our experiments, we made an attempt at introducing symmetry constraints explicitly, but this had an adverse effect of slowing down the solver. We leave it for future work to address both solver specialization and symmetry constraints.

REFERENCES

[1] A. Adadi and M. Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)," IEEE Access, vol. 6, pp. 52138–52160, 2018.

[2] A. A. Freitas, "Comprehensible classification models," ACM SIGKDD Explorations Newsletter, vol. 15, no. 1, pp. 1–10, 2014.

[3] A. Holzinger, C. Biemann, Pattichis, C.S. and Kell, D.B.,, "What do we need to build explainable AI systems for the medical domain?," 2017. [Online]. Available: https://arxiv.org/abs/1712.09923.

[4] F. K. Dosilovic, M. Brcic, and N. Hlupic, "Explainable artificial intelligence: A survey," 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018.

[5] D. Rodriguez, R. Ruiz, J. Riquelme, and J. Aguilar-Ruiz, "Searching for rules to detect defective modules: A subgroup discovery approach," Information Sciences, vol. 191, pp. 14–30, 2012.

[6] European Commission, "White Paper on Artificial Intelligence: a European approach to excellence and trust," 2020. [Online]. Available: https://ec.europa.eu/info/sites/info/files/commission-white-paper-artificial-intelligence-feb2020_en.pdf. [Accessed: 11-May-2020].

[7] Council of European Union, "Consolidated text: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)," 2016. [Online]. Available: http://data.europa.eu/eli/reg/2016/679/2016-05-04. [Accessed: 11-May-2020].

[8] M. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, 2016.

[9] D. M. Malioutov, K. R. Varshney, A. Emad, and S. Dash, "Learning Interpretable Classification Rules with Boolean Compressed Sensing," Studies in Big Data Transparent Data Mining for Big and Small Data, pp. 95–121, 2017.

[10] G. Su, D. Wei, K. R. Varshney, and D. M. Malioutov, "Learning sparse two-level boolean rules," 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), 2016.

[11] D. Malioutov, K. Varshney, "Exact rule learning via boolean compressed sensing," Conference on Machine Learning, pp. 765–773, 2013.

[12] A. Ignatiev, F. Pereira, N. Narodytska, and J. Marques-Silva, "A SAT-Based Approach to Learn Explainable Decision Sets," Automated Reasoning Lecture Notes in Computer Science, pp. 627–645, 2018.

[13] D. Malioutov and K. S. Meel, "MLIC: A MaxSAT-Based Framework for Learning Interpretable Classification Rules," Lecture Notes in Computer Science Principles and Practice of Constraint Programming, pp. 312–327, 2018.

[14] B. Ghosh and K. S. Meel, "IMLI: An incremental framework for MaxSAT-based learning of interpretable classification rules," Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, 2019.

[15] R. L. Rivest, "Learning decision lists," Machine Learning, vol. 2, no. 3, pp. 229–246, 1987.
[16] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable Decision Sets," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
[17] L. Bottou and V. Vapnik, "Local Learning Algorithms," Neural Computation, vol. 4, no. 6, pp. 888–900, 1992.
[18] "Locally Weighted Learning," Encyclopedia of Machine Learning and Data Mining, pp. 759–759, 2017.
[19] J.H. Friedman, R. Kohavi, and Y. Yun, "August. Lazy decision trees," AAAI/IAAI, Vol. 1, pp. 717–724, 1996.
[20] T. Nguyen and S. Sanner. "Algorithms for direct 0-1 loss optimization in binary classification," International Conference on Machine Learning, pp. 1085-1093, 2013.
[21] A. Boudane, S. Jabbour, B. Raddaoui, and L. Sais, "Efficient SAT-Based Encodings of Conditional Cardinality Constraints."
[22] T. Nguyen, S. Sanner, "Determination of the spread parameter in the Gaussian kernel for classification and regression," Neurocomputing, vol. 55, no. 3-4, pp. 643-663, 2003.
[23] M. Piotrow, "UWrMaxSat-a new MiniSat+-based Solver," MaxSAT Evaluation 2019, pp. 11, 2019.
[24] L. Breiman, J. Friedman, C. Stone, R. Olshen, "Classification and regression trees," CRC press, 1984.
[25] W. Cohen, "Fast Effective Rule Induction," Morgan Kaufmann, Twelfth International Conference on Machine Learning, pp. 115-123, 1995.
[26] F. Eibe, M. Hall, and W. Ian, The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.
[27] P. Langley, "Elements of machine learning," Morgan Kaufmann, 1996.

HENRIK CAO received his B.A.Sc. degree in bioinformation technology in 2016 and his M.A.Sc degree in complex systems in 2019 from Aalto University, Espoo, Finland, where he is currently pursuing his Ph.D. degree in the Department of Computer Science.

From 2019 to 2020 he worked as a consultant with the Social Insurance Institution of Finland. He has completed internships at Nippon Paper Industries, Tokyo, Japan, in 2013 and at Aalto University in 2018. His research interests include propositional satisfiability, answer set programming and the application of constraint solvers in artificial intelligence.

RIKU SARLIN received his M.Sc. degree in Helsinki University of Technology in 1997, majoring in work psychology and computer science. He has done a career in software engineering, and currently works as a Chief Architect for Social Insurance Institution of Finland. His work interests include applying AI in the process of preparing for and making administrative decisions for social benefits.

ALEXANDER JUNG has obtained a Phd (Sub auspiciis Praesidentis) in statistical signal processing from TU Vienna in 2012. He has been Post-Doc at ETH Zurich and TU Vienna before he joined Aalto University as Assistant Professor for Machine Learning in 2015. The focus of his research is to understand fundamental limits and efficient methods for machine learning problems arising in various application domains. The quality of his research has been recognized by a Best Student Paper Award at the conference IEEE ICASSP 2011 and an Amazon Web Services Machine Learning Award in 2018. He co-authored a paper that was finalist for the best student paper award at Asilomar 2017. While at Aalto University, he has redesigned the main course on Machine Learning and developed a new online course "Machine Learning with Python". He has been elected as Teacher of the Year 2018 by the department of Computer Science at Aalto University. He serves as the chair of the Signal Processing and Circuits & Systems Chapter within the IEEE Finland Section.