SAT-Based Learning of Compact Binary Decision Diagrams for Classification

Pouya Shati
Department of Computer Science, University of Toronto, Canada
Vector Institute, Toronto, Canada

Eldan Cohen
Department of Mechanical and Industrial Engineering, University of Toronto, Canada

Sheila McIlraith
Department of Computer Science, University of Toronto, Canada
Vector Institute, Toronto, Canada

Abstract

Decision trees are a popular classification model in machine learning due to their interpretability and performance. However, the number of splits in decision trees grow exponentially with their depth which can incur a higher computational cost, increase data fragmentation, hinder interpretability, and restrict their applicability to memory-constrained hardware. In constrast, binary decision diagrams (BDD) utilize the same split across each level, leading to a linear number of splits in total. Recent work has considered optimal binary decision diagrams (BDD) as compact and accurate classification models, but has only focused on binary datasets and has not explicitly optimized the compactness of the resulting diagrams. In this work, we present a SAT-based encoding for a multi-terminal variant of BDDs (MTBDDs) that incorporates a state-of-the-art direct encoding of numerical features. We then develop and evaluate different approaches to explicitly optimize the compactness of the diagrams. In one family of approaches, we learn a tree BDD first and model the size of the diagram the tree will be reduced to as a secondary objective, in a one-stage or two-stage optimization scheme. Alternatively, we directly learn diagrams that support multi-dimensional splits for improved expressiveness. Our experiments show that direct encoding of numerical features leads to better performance. Furthermore, we show that exact optimization of size leads to more compact solutions while maintaining higher accuracy. Finally, our experiments show that multi-dimensional splits are a viable approach to achieving higher expressiveness with a lower computational cost.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Computing methodologies → Machine learning

Keywords and phrases Binary Decision Diagram, Classification, Compactness, Numeric Data, MaxSAT

Digital Object Identifier 10.4230/LIPIcs.CP.2023.33

Supplementary Material *Software (Source code)*: `https://github.com/PouyaShati/BDD`

Funding We gratefully acknowledge funding from Natural Sciences and Engineering Research Council of Canada (NSERC) and the CIFAR AI Chairs program (Vector Institute).

1 Introduction

Classifiers are complete functions for assigning labels to datapoints, learned from a limited set of supervised training data. A classifier is trained to focus on informative aspects of the input and extract patterns from the data to make decisions. In complex black-box classifiers such as deep and convoluted neural networks, it is often challenging to understand the influential features of the data and the inner-workings of the decision-making [36, 38]. In contrast, interpretable classifiers such as decision trees [1, 26, 33, 34] and diagrams [16, 24] are concise and easy to understand [10]. The simplicity in interpretable solutions makes them the

perfect candidates for when we need to formally analyze or explain the classifier's behavior [27, 19, 20]. Surprisingly, the benefit of interpretability does not come with significant cost to accuracy in many applications [31, 25].

Binary decision diagrams (BDD) are graph representations of functions with binary inputs and are widely used in logical synthesis and formal verification methods [8, 2, 28, 23]. While equivalent to truth tables in purpose, BDDs are more compact since redundant sub-tables can be eliminated and merged [23]. A multi-terminal BDD (MTBDD) is a BDD variant that supports multiple outputs [11].

Decision trees are the most common form of interpretable classifiers (e.g., [7, 29, 30, 3, 37, 18, 5, 33, 34]). However, the number of splits double at each level of a decision tree, making it challenging to interpret the solution as depth increases [15]. The large number of splits in decision trees can hinder the learning performance given that the search space grows exponentially with each level. Further, deep splits that only affect a small number of datapoints can cause data fragmentation and overfitting [35, 21, 14]. Lastly, the exponential number of splits in decision trees restricts their applicability to memory-constrained hardware [35]. The sequential nature of decision-making in BDDs resembles that of decision trees, but unlike decision trees, the same split is used across a level, leading to a linear number of splits.

In this paper, we choose BDDs as our interpretable classifiers in order to emphasize compactness to an even greater degree compared to popular interpretable approaches, e.g. ones based on decision trees. Alongside the primary objective of accuracy, the size of a BDD classifier is also encoded and optimized. Our encoding for learning BDDs is inspired by the encoding of numerical branchings in Shati et al. [33, 34], which allows us to directly learn splits over numeric features without explicitly binarizing them. We then expand the notion of learning splits to multiple dimensions in order to learn solutions from a limited but distinctively interpretable family of diagrams.

The main contributions of this paper are as follows:

1. We present a novel SAT encoding of maximum accuracy BDD classifiers for numerical datasets. Our model represents a non-reduced BDD which can be reduced according to its sequence of terminals.
2. We extend our encoding by modelling the size of the final reduced BDD. The size encoding can be used as a secondary weighted objective, or it can be optimized in a second stage.
3. We present a variant of our encoding which supports direct learning of diagrams through multi-dimensional splits. Multi-dimensional BDDs enable learning more efficiently as they provide more expressive solutions within the same number of splits.
4. We run extensive experiments which demonstrate that our base encoding outperforms the state of the art in runtime and accuracy. Additionally, we show that explicitly optimizing size leads to significantly more compact solutions while maintaining high accuracy. Finally, we show that multi-dimensional BDDs scale better than BDDs due to their expressiveness and the size of their encoding.

2 Related Work

Decision tree classifiers are traditionally constructed via local search and heuristics [7, 29, 30]. Recent advances in tools and techniques for exact optimization have enabled approaches for finding optimal decision trees via branch-and-bound search [1], SAT-based encodings [33, 18, 3], Constraint Programming [37], or Mixed Integer Programming [5]. Exact optimization produces solutions that are optimal in accuracy, size, or both [33]. The size of a decision tree is often measured by its depth or number of nodes, which does not take into account the amount

of redundancy between nodes. Decision diagrams address redundancy directly by merging equivalent nodes. Approaches to learn optimal decision diagrams via exact optimization usually require a pre-determined skeleton as input. For example, Florio et al. [14] requires user-provided width for each level, limiting solutions but guaranteeing compactness and improving performance. Oblivious decision diagrams encourage compactness even further by requiring all splits of the same level to be the same, leading to only a linear number of splits instead of exponential as in ordinary decision trees.

Binary Decision Diagrams (BDD) are oblivious decision diagrams that can be reduced and ordered, making them ideal candidates to represent boolean functions. While BDDs are commonly used for hardware synthesis [8, 2, 28, 23], there has been a recent focus on utilizing BDD classifiers as interpretable solutions [16, 9]. For example, Hu et al. [16] uses MaxSAT to learn a maximum accuracy tree BDD which will then be reduced into a diagram. On the other hand, Cabodi et al. [9] considers a SAT-based approach for learning BDDs of minimum size that correctly classify all the training data.

3 Background

3.1 Binary Decision Diagrams

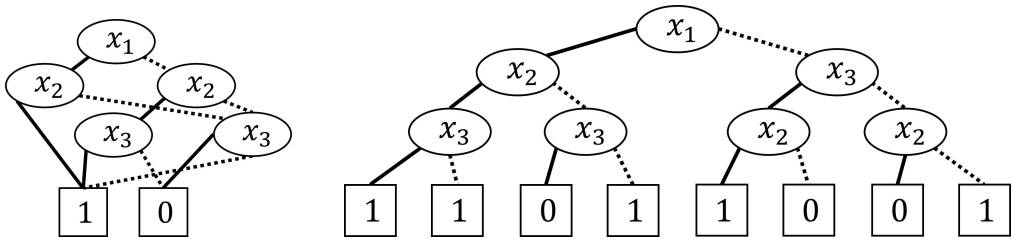
In this section, we first define standard Binary Decision Diagrams which can be used for binary classification. To support multi-class classification, we extend the BDD definition to its multi-terminal variant.

▶ **Definition 1** (Binary Decision Diagram). *Given a boolean function operating on boolean input, a Binary Decision Diagram (BDD) is a graph representation of the function. A BDD is a rooted, directed, and acyclic graph with sink nodes as terminals ($\mathcal{N}_T$) and the rest as decision nodes ($\mathcal{N}_D$). Each decision node is labelled with a split and has two outgoing edges, corresponding to the two (1/0) possible values of the boolean input. The terminal nodes are assigned output values* 0 *or* 1*. The size of a BDD is the number of its decision nodes.*

▶ **Definition 2** (Multi-Terminal Binary Decision Diagram). *Given a function operating on binary features with constant range* $[1..k]$*, a Multi-Terminal Binary Decision Diagram (MTBDD) is a graph representation of the function. An MTBDD is a BDD with its terminals supporting* k *output values instead of* 2*.*

Throughout the paper, we will simply refer to MTBDDs as BDDs to be more concise. We adopt standard BDD terminology with minor exceptions. As a concise way of addressing the nodes, we borrow from the decision tree literature to view the outgoing edges as parent-child relationships. Specifically, we address the node connected via the 1 (0) outgoing edge as the left (right) child. Moreover, we refer to the boolean inputs assigned to decision nodes as splits rather than features or variables, to avoid confusion with the features of our data and the variables of our encoding. Finally, considering that the features in our data are numerical rather than binary, we overload the definition of splits to represent pairs of numerical features and a valid threshold.

▶ **Definition 3** (Split). *Given a finite set of numerical features F, a split is a binarization of a numerical feature through pairing with a threshold value $(f, \alpha) \in F \times \mathbb{R}$. A split assigns a value of* 1 *to each point x that comes before the threshold ($x[f] \leq \alpha$). Similarly, it assigns a value of* 0 *to each point x that comes after the threshold ($x[f] > \alpha$).*



(a) Ordered and reduced. **(b)** Non-ordered and non-reduced.

Figure 1 Two BDDs representing the same boolean function. Solid (dotted) lines correspond to value 1 (0).

In order to calculate the value of a function for a given input using its BDD representation, we start from the root, move to the left (right) child if the input has value 1 (0) for the current split, and assign an output when a terminal is reached. A node is said to *contain* an input if the node is on the input's path from the root to a terminal.

The same function with binary splits can be represented with multiple BDDs. In order to uniquely represent a function, we define the BDD properties of being *ordered* and *reduced*. The ordered property enforces the same sequence of splits along each path, making BDDs practically equivalent to Oblivious Read-Once Decision Graphs (OODGs). The reduced property requires all equivalent parts of the BDD to be merged and all of the redundant parts to be removed. Given a function and an ordering of splits, there only exists one ordered and reduced BDD representing the function.

▶ **Definition 4** (Ordered BDD and Split Sequence). *A BDD is said to be ordered if the splits observed along every path from the root to a terminal respect a singular total ordering, called its split sequence.*

▶ **Definition 5** (Node Equivalency and Redundancy). *Given a binary decision diagram $\mathcal{T}$ and two of its nodes $t_1, t_2 \in \mathcal{N}$, t_1 and t_2 are equivalent if they are both decision nodes with the same split and the sub-graph of t_1 and its descendants is isomorphic to the sub-graph of t_2 and its descendants, or if they are both terminal nodes with the same label. Furthermore, a decision node $t \in \mathcal{N}_D$ is redundant if its left and right children are equivalent.*

▶ **Definition 6** (Reduced BDD). *A BDD is said to be reduced if all of its equivalent nodes are merged and all of its redundant nodes are replaced with their children.*

We assume a BDD to be ordered and reduced unless noted otherwise. Figure 1 depicts two BDDs representing the same boolean function, one ordered and reduced and the other one not. It has been shown in the literature that independent of the merging and elimination process, the final result of reducing a BDD is always the same. We formally state the uniqueness of reduced BDDs in Proposition 7 and refer the reader to the BDD literature for more details [23].[1]

▶ **Proposition 7.** *Every function operating on splits and a given split sequence is represented by exactly one ordered and reduced BDD up to renaming.*

[1] Note that when employing standard BDD terminology, redundant and equivalent nodes are defined through binary sequences called *beads*. Beads correspond to nodes of the uniquely reduced BDD described in Proposition 7.

3.2 Weighted Partial MaxSAT

In this section, we describe the MaxSAT paradigm which we use to learn binary decision diagrams. A SAT formula is a conjunction of clauses, where each clause is a disjunction of literals, and each literal is either a Boolean variable or its negation. In a weighted partial MaxSAT instance, clauses are categorized into **hard** and **weighted soft** clauses. The goal is then to find a truth assignment to variables which satisfies all of the hard clauses and maximizes the total weight of the satisfied soft clauses.

4 MaxSAT Encoding of BDD Classifiers

In this section, we propose a MaxSAT encoding to find a BDD classifier with maximum accuracy. The inputs to our problem are: a training dataset X over a set of numerical features F and labels K, a ground-truth label y_i for each point $x_i \in X$, and a maximum number of splits s_{max}. The outputs are s_{max} splits $\theta \in \{F \times \mathbb{R}\}^{s_{max}}$ with terminal labelling $\gamma : \{0,1\}^{s_{max}} \mapsto K$. Similar to previous work of Hu et al. [16], we learn an ordered yet non-reduced (tree) BDD first. We then focus on merging and replacing nodes towards a reduced BDD. Inspired by the numerical branching in Shati et al. [33, 34], we encode splits (Definition 3) without the need for prior binarization of data which was shown to lead to significant performance degradation in decision trees. We instead directly encode how each split directs each point, while making sure the order of values is respected given a selected feature.

4.1 Variables

The following set of binary variables represents different aspects of modeling splits, labels, and accuracy in our BDD encoding.

- $a_{s,j}$: The feature chosen at split s is or comes before j.
- $d_{s,i}$: Point x_i is directed to the left child at split s.
- $c_{t,l}$: Output label l is assigned to terminal node t.
- o_i: Point x_i is classified correctly.

4.2 Clauses

We propose the following sets of Conjunctive Normal Form (CNF) clauses for modelling a non-reduced ordered BDD. The clauses in Eqs. (1-2) guarantee that one feature is selected at each split by enforcing the ordered encoding of $a_{s,j}$ variables. The clauses in Eqs. (3-5) guarantee that for each split and chosen feature, the points directed to the left (resp. right) have a lesser (resp. greater) value compared to a threshold. The clauses in Eqs. (6-7) guarantee that one output label is chosen for each terminal node. The clauses in Eq. (8) guarantee that a point can only be considered classified if it ends up in a terminal node with the same label as its ground truth. We use $O_j(X)$ to denote the set of all consecutive pairs when the members of X are sorted based on their j values, $O_j^{=}(X)$ to denote the subset of pairs with equal j values, i.e., $O_j(X) \cap \{(i_1, i_2) \,|\, x_{i_1}[j] = x_{i_2}[j]\}$, and $\#_j^1$ to denote the index of the point with the smallest j value. Furthermore, we set $a_{s,|F|}$ to the false constant and define $A_R(t)$ ($A_L(t)$) as the set of right (left) ancestors of terminal t.

$$(a_{s,j}, \neg a_{s,j+1}) \qquad s < s_{max}, j \in F \qquad (1)$$

$$(a_{s,0}) \qquad s < s_{max} \qquad (2)$$

$$(\neg a_{s,j}, a_{s,j+1}, d_{s,i_1}, \neg d_{s,i_2}) \qquad s < s_{max}, j \in F, (i_1, i_2) \in O_j(X) \qquad (3)$$

$$(\neg a_{s,j}, a_{s,j+1}, \neg d_{s,i_1}, d_{s,i_2}) \qquad s < s_{max}, j \in F, (i_1, i_2) \in O_j^{=}(X) \qquad (4)$$

$$(\neg a_{s,j}, a_{s,j+1}, d_{s,\#_j^1}) \qquad s < s_{max}, j \in F \qquad (5)$$

$$(\neg c_{t,l_1}, \neg c_{t,l_2}) \qquad t \in \mathcal{N}_T, l_1, l_2 \in K \qquad (6)$$

$$(\bigvee_{l \in L} c_{t,l}) \qquad t \in \mathcal{N}_T \qquad (7)$$

$$(\bigvee_{s \in A_L(t)} \neg d_{s,i}, \bigvee_{s \in A_R(t)} d_{s,i}, c_{t,y_i}, \neg o_i) \qquad t \in \mathcal{N}_T, x_i \in X \qquad (8)$$

In order to model the objective, we include the soft clauses in Eq. (9) with unit weights, which represent correctly classifying as many training points as possible.

$$(o_i) \qquad x_i \in X \qquad (9)$$

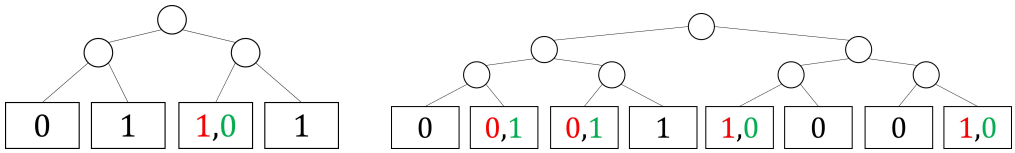
4.3 Decoding

An assignment to the variables in Section 4.1 which is satisfying with regard to the hard clauses in Section 4.2 is decoded into a reduced BDD in two steps. First, the assignment is decoded into a non-reduced BDD. Specifically, the labels of terminals are decoded from the selected labels ($c_{t,l}$) and the sequence of splits are decoded from the pairings of selected features ($a_{s,j}$ values) and thresholds ($d_{s,i}$ values). Second, the resulting BDD is reduced by merging equivalent nodes and eliminating redundant nodes.

5 Size Optimization

In the encoding presented in Section 4, not all terminals are guaranteed to contain training points. Thus, we may end up with empty terminals, which we can relabel without affecting the training accuracy. However, the labels of such terminals can affect the prediction on unseen datapoints as well as the size of the final reduced BDD. In Hu et al. [16], the labels of empty terminals are decided arbitrarily by the solver and they investigate the impact of two post-processing relabelling heuristics on the testing accuracy (i.e., accuracy on unseen data). The first heuristic assigns the majority label of the terminal's first non-empty ancestor. The second heuristic finds and merges equivalent nodes in a greedy top-down search. Hu et al.'s experiments showed that the heuristics do not have significant impact on the testing accuracy. However, as we demonstrate in Figure 2a and Figure 2b, such heuristic approaches can increase the size of the final reduced BDDs.

In this section, we propose to use exact optimization for solving the same problem as in Section 4, with additional compactness considerations via deciding the labels of empty terminals. Specifically, we aim to model the size of the BDD after the merging of its equivalent nodes and the removal of its redundant nodes. The variables and clauses introduced in this section can either be added to the encoding in Section 4 as a secondary objective (i.e. 1-stage approach), or they can be used as a separate post-processing stage alongside variables $c_{t,l}$ and the clauses in Eqs. (6-7) with the labels of non-empty terminals fixed (i.e. 2-stage approach).



(a) Parent Majority. **(b)** Greedy Subgraph Merge.

Figure 2 Examples for heuristics of assigning labels to empty terminals where size is increased. Black labels represent non-empty terminals, red labels are chosen by the heuristics, and green labels are the original labels which lead to a more compact solution.

5.1 Variables

The following set of binary variables represents uniqueness and repetition aspects of terminal labels, required for modelling the size of the ultimately reduced BDD.

- σ_{t_1,t_2}: Terminals t_1 and t_2 have been assigned different output labels.
- $b_{t,\Delta}$: The sequence of Δ labels starting from terminal node t (inclusive) cannot be divided into two equal sub-sequences.
- $r_{t_1,t_2,\Delta}$: The sequence of Δ labels starting from terminal node t_1 is equal to the sequence of Δ labels starting from terminal node t_2 (both inclusive).

Note that the terminal order which is referred to in the variable definitions, is the order of appearance in the depth-first search of the tree with the left outgoing edges prioritized. We say a sequence of terminals correspond to a decision node, if they are the sequence of terminals in the decision node's sub-graph. Moreover, we say a single terminal coincides with a decision node if it marks the beginning the node's sequence of terminals.

Note that variables σ_{t_1,t_2}, $b_{t,\Delta}$, and $r_{t_1,t_2,\Delta}$ are not defined for all possible index combinations. Specifically, Δ always refers to the length of a sequence of terminals corresponding to a decision node (a power of 2), and can uniquely determine the node, if paired with a coinciding terminal. Consequently, $b_{t,\Delta}$ is only used to represent that the decision node at level $s_{max} - log_2(\Delta) + 1$ coinciding with t is not redundant. Moreover, $r_{t_1,t_2,\Delta}$ is only used to represent that the two decision nodes at level $s_{max} - log_2(\Delta) + 1$ coinciding with t_1 and t_2 are equivalent. Lastly, σ_{t_1,t_2} is only used when it affects the equivalency of two decision nodes.

5.2 Clauses

We first define two sets to help with the clause construction. The set $P(s_{max})$ contains all possible lengths of terminal sequences corresponding to decision nodes except for the root (i.e., all $2^{p'}$ where $p' < s_{max}$). The set $G(s_{max})$ contains all triples (t_1, t_2, Δ) where comparing the labels of terminals t_1 and t_2 has impact on a decision node at level $s_{max} - log_2(\Delta) + 1$ being redundant.

$$
\begin{aligned}
&G(1) = \{(0, 1, 2)\} \\
&G(p) = G(p-1) \cup \{(t_1 + 2^{p-1}, t_2 + 2^{p-1}, \Delta) | (t_1, t_2, \Delta) \in G(p-1)\} \\
&\qquad \cup \{(t, t + 2^{p-1}, 2^p) | 0 \leq t < 2^{p-1}\}
\end{aligned}
$$

We propose the following sets of CNF clauses for modelling the size of a reduced BDD given its terminal labels. The clauses in Eqs. (10-12) guarantee that σ_{t_1,t_2} variables correctly represent the difference in labels. The clauses in Eq. (13) guarantee that each decision node

can only be redundant if all of the corresponding terminal pairs have the same label. The clauses in Eqs. (14-15) guarantee that a pair of decision nodes can only be equivalent if their corresponding sequence terminals have the same labels.

$$(\neg c_{t_1,l}, \neg c_{t_2,l}, \neg \sigma_{t_1,t_2}) \qquad (t_1, t_2, \Delta) \in G(s_{max}), l \in K \tag{10}$$
$$(\neg c_{t_1,l}, c_{t_2,l}, \sigma_{t_1,t_2}) \qquad (t_1, t_2, \Delta) \in G(s_{max}), l \in K \tag{11}$$
$$(c_{t_1,l}, \neg c_{t_2,l}, \sigma_{t_1,t_2}) \qquad (t_1, t_2, \Delta) \in G(s_{max}), l \in K \tag{12}$$
$$(b_{\Delta \lfloor t_1/\Delta \rfloor, \Delta}, \neg \sigma_{t_1,t_2}) \qquad (t_1, t_2, \Delta) \in G(s_{max}) \tag{13}$$
$$(\neg r_{t_1\Delta,t_2\Delta,\Delta}, \neg c_{t_1\Delta+\delta,l}, c_{t_2\Delta+\delta,l}) \qquad \Delta \in P(s_{max}), t_1 < t_2 < 2^{s_{max}}/\Delta, \delta < \Delta, l \in K \tag{14}$$
$$(\neg r_{t_1\Delta,t_2\Delta,\Delta}, c_{t_1\Delta+\delta,l}, \neg c_{t_2\Delta+\delta,l}) \qquad \Delta \in P(s_{max}), t_1 < t_2 < 2^{s_{max}}/\Delta, \delta < \Delta, l \in K \tag{15}$$

To model our objective, we include the soft clauses in Eq. (16) which represent each decision node to be either redundant or equivalent to a previous one. Maximizing the number of redundant of equivalent decision nodes will consequently minimize the size of our reduced BDD. Note that if size is being considered as a secondary objective to accuracy, the clauses in Eq. (16) can be weighted against the clauses in Eq. (9) proportionately. Otherwise, if size is our second-stage objective, we can use the clauses in Eq. (16) with unit weights.

$$(\bigvee_{0 \leq t_2 < t} r_{t_2\Delta,t\Delta,\Delta}, \neg b_{t\Delta,\Delta}) \qquad \Delta \in P(s_{max}), t < 2^{s_{max}}/\Delta \tag{16}$$

5.3 Decoding

We first show the soundness of the size encoding.

▶ **Proposition 8.** *Given a sequence of terminal labels $c_{t,l}$, the encoding in Sections (5.1,5.2) has an optimal objective value equal to the reduced size of an ordered tree BDD with the given terminals.*

Proof Sketch. Given the alternative interpretation of variables described in Section 5.2, we can conclude that the objective clauses in Eq. (16) aim to minimize the number of decision nodes that are not redundant and not equivalent to any of the nodes that come before them in the same level. Considering that equivalent decision node pairs can only appear in the same level, we can restate the objective as minimizing the number of decision nodes that remain when all of equivalent ones are merged and redundant ones are eliminated. According to Proposition 7, this objective will lead to a unique reduced BDD in its most compact form, proving that our encoding correctly models and minimizes the size of our solution. ◀

To decode the solution, we simply need to do as we did in Section 4.3, since the additional variables for size encoding are not involved in structuring the BDD. Once we decode and reduce the solution as before, we will end up with one decision node for every unsatisfied soft clause in Eq. (16).

6 MaxSAT Encoding of Multi-Dimensional BDDs

In Section 4 and Section 5, we consider learning BDDs by finding trees and reducing them to diagrams. However, the clauses in Eq.(8) are exponential in the number of splits and multiplied by the size of the dataset. Learning diagrams directly can help us consider more expressive solutions without increasing the number of splits, resulting in smaller encodings. In this section, we look at a family of diagrams that contain splits over multiple features at each level, i.e., multi-dimensional splits which themselves are BDDs with two labels.

▶ **Definition 9** (Directional Inner BDD). *Given a finite set of numerical features F and a dimension D, a directional inner BDD $\mathcal{T}^{\leftrightarrow}$ is a BDD operating on D splits $\theta^{\leftrightarrow} \in \{F \times \mathbb{R}\}^D$ with* 0 *and* 1 *as labels.*

A multi-dimensional BDD operates on multi-dimensional splits. A multi-dimensional split uses a directional inner BDD to direct points towards left (label 1) and right (label 0), rather than using a single split.

▶ **Problem 10** (Multi-dimensional BDD Learning Problem). *Given a finite set of numerical features F, a finite set of labels K, a set of training points $x_i \in X$ with corresponding labels $y_i \in K$, a sequence of dimensions $\mathcal{D} = \{D_0, D_1, D_{s_{max}-1}\}$, and a number of multi-dimensional splits s_{max}, the goal is to find a sequence of directional inner BDDs $\theta^M = \{\mathcal{T}_0^{\leftrightarrow}, \mathcal{T}_1^{\leftrightarrow}, ..., \mathcal{T}_{s_{max}-1}^{\leftrightarrow}\}$ of respective dimensions $D_0, D_1, ..., D_{s_{max}-1}$ with terminal labelling $\gamma^M : \{0,1\}^{s_{max}} \mapsto K$ to construct a BDD with the directional inner BDDs as splits. The objective for learning Multi-dimensional BDDs is high accuracy.*

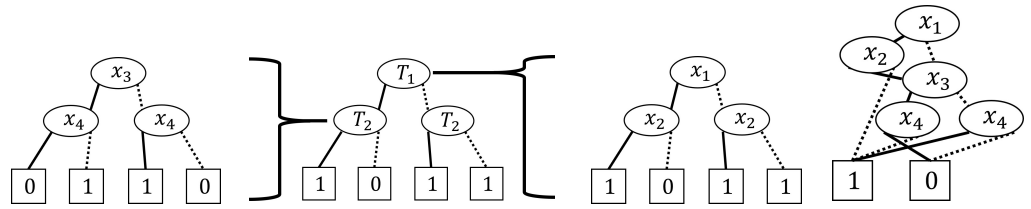
A multi-dimensional split is a generalization of an ordinary split, which makes a multi-dimensional BDD more expressive compared to an ordinary BDD with the same number of splits. Previous works have considered other representations that are designed to be more expressive than BDDs, such as sentential decision diagrams [12] or read-k-times branching programs [6, 22]. Next, we formally compare the expressiveness of BDDs against their multi-dimensional variants in the other direction.

▶ **Theorem 11.** *Consider a multi-dimensional BDD $\mathcal{T}^M$ operating on s_{max} directional inner BDDs $\mathcal{T}_s^{\leftrightarrow}$ with split sequences $\theta_s^{\leftrightarrow}$ and terminal labellings $\gamma_s^{\leftrightarrow}$, which itself has a terminal labelling γ^M. The binary function that $\mathcal{T}^M$ represents is the same as the binary function represented by a BDD $\mathcal{T}$ with split sequence $\theta(\#(s,h)) = \theta_s^{\leftrightarrow}(h)$ and terminal labelling $\gamma(t) = \gamma^M(\gamma_0^{\leftrightarrow}(t_0)\gamma_1^{\leftrightarrow}(t_1)...\gamma_{s_{max}-1}^{\leftrightarrow}(t_{s_{max}-1}))$ where $\#(s,h)$ produces a complete ordering of S_H by concatenating the split sequences of each directional inner BDD, and t_0, t_1, ..., $t_{s_{max}-1}$ are sub-sequences of t divided based on the sequence of dimensions.*

Proof Sketch. The BDD using multi-dimensional splits can be transformed into a tree BDD with ordinary splits. In the transformation, every split of dimension D will correspond to D levels where nodes are expanded 2^D-fold. With each expansion, we add annotations to the nodes specifying the label which they were assigned to by the corresponding directional inner BDD. Once all multi-dimensional splits are added, the final level of nodes are considered as terminals and are labelled according to how γ^M labels their annotations. Note that the resulting tree can be turned into a diagram by merging all of the nodes that have equal annotations and replacing all of redundant nodes with their children. ◀

We can conclude from Theorem 11, that multi-dimensional BDDs operating on dimensions $\{D_0, D_1, ..., D_{s_{max}-1}\}$ with $D_{total} = \sum_{i<s_{max}} D_i$ are less expressive than BDDs operating on D_{total} splits. Combining with the aforementioned fact that multi-dimensional BDDs operating on $\{D_0, D_1, ..., D_{s_{max}-1}\}$ dimensions are trivially more expressive than BDDs operating on s_{max} splits, we have shown a lower and upper bound number of BDD splits when discussing the expressiveness of a multi-dimensional BDD.

Figure 3 shows how a multi-dimensional BDD can operate on two directional inner BDDs and how it can be unfolded to operate on ordinary splits.



(a) $\mathcal{T}_2^{\leftrightarrow}$. **(b)** $\mathcal{T}^M$. **(c)** $\mathcal{T}_1^{\leftrightarrow}$. **(d)** Unfolded $\mathcal{T}^M$.

Figure 3 Multi-dimensional BDD $\mathcal{T}^M$ operating on directional inner BDDs $\mathcal{T}_1^{\leftrightarrow}$ and $\mathcal{T}_2^{\leftrightarrow}$ and its unfolded version.

6.1 Variables

In order to encode a multi-dimensional BDD, we use the same set of variables as Section 4.1 but remove $a_{s,j}$ variables since multi-dimensional splits require more features to be selected at each split. Furthermore, we add the following variables to encode the inner-workings of each directional inner BDD.

- $\widehat{a}_{(s,h),j}$: The feature chosen at split h of directional inner BDD s is or comes before j.
- $\widehat{d}_{(s,h),i}$: Point x_i is directed to left at split h of directional inner BDD s.
- $\widehat{c}_{s,t}$: Terminal t of directional inner BDD s is assigned the label 1.

6.2 Clauses

We discard clauses Eqs. (1-6) from the original encoding since splits are learned differently than before. We keep the clauses Eqs. (6-9) however, since we still need the labels and the appearance of points at terminals to be modelled correctly. Lastly, we add CNF clauses for modelling multi-dimensional splits, in order to complete our encoding of a multi-dimensional BDD.

The clauses in Eqs. (17-18) guarantee that one feature is selected at each split of each directional inner BDD by enforcing the ordered encoding of $\widehat{a}_{(s,h),j}$ variables. The clauses in Eqs. (19-21) guarantee that each split of each directional inner BDD conforms to a pairing of selected feature and threshold. The clauses in Eqs. (22-23) guarantee that the direction of a point in a multi-dimensional split matches the label of its containing leaf in the corresponding directional inner BDD. The clauses in Eq. (24) guarantee the leftmost leaf in each directional inner BDD to be assigned the 0 label (analogous to Eq. (5), see Shati et al. [33, 34] for further discussion). The set S_H contains all pairs of directional inner BDDs and their corresponding splits $S_H = \{(s,h) \,|\, s < s_{max}, h < D_s\}$, $\mathcal{N}_T^s$ contains the terminals of directional inner BDD s, $A_R(s,t)$ ($A_L(s,t)$) contains the right (left) ancestors of terminal t within directional inner BDD s, and $\widehat{a}_{(s,h),|F|}$ is set to the false constant.

$$(\widehat{a}_{(s,h),j}, \neg\widehat{a}_{(s,h),j+1}) \qquad (s,h) \in S_H, j \in F \quad (17)$$

$$(\widehat{a}_{(s,h),0}) \qquad (s,h) \in S_H \quad (18)$$

$$(\neg\widehat{a}_{(s,h),j}, \widehat{a}_{(s,h),j+1}, \widehat{d}_{(s,h),i_1}, \neg\widehat{d}_{(s,h),i_2}) \qquad (s,h) \in S_H, j \in F, (i_1,i_2) \in O_j(X) \quad (19)$$

$$(\neg\widehat{a}_{(s,h),j}, \widehat{a}_{(s,h),j+1}, \neg\widehat{d}_{(s,h),i_1}, \widehat{d}_{(s,h),i_2}) \qquad (s,h) \in S_H, j \in F, (i_1,i_2) \in O_j^{=}(X) \quad (20)$$

$$(\neg\widehat{a}_{(s,h),j}, \widehat{a}_{(s,h),j+1}, \widehat{d}_{(s,h),\#_j^1}) \qquad (s,h) \in S_H, j \in F \quad (21)$$

$$(\bigvee_{h \in A_R(s,t)} \widehat{d}_{(s,h),i}, \bigvee_{h \in A_L(s,t)} \neg\widehat{d}_{(s,h),i}, d_{s,i}, \neg\widehat{c}_{s,t}) \qquad s \in S, x_i \in X, t \in \mathcal{N}_T^s \quad (22)$$

$$\left(\bigvee_{h\in A_R(s,t)} \widehat{d}_{(s,h),i}, \bigvee_{h\in A_L(s,t)} \neg\widehat{d}_{(s,h),i}, \neg d_{s,i}, \widehat{c}_{s,t}\right) \qquad s \in S, x_i \in X, t \in \mathcal{N}_T^s \quad (23)$$

$$(\widehat{c}_{s,0}) \qquad s \in S \quad (24)$$

6.3 Decoding

A satisfying assignment to the variables in Section 6.1 with regard to the clauses in Section 6.2, can be decoded into a tree BDD operating on multi-dimensional splits. In order to further transform the results into a BDD operating on 1-dimensional splits, we unfold the multi-dimensional splits and multiply the terminals according to the proof sketch for Theorem 11. Note that the number of splits in the resulting BDD is equal to the sum of dimensions from the multi-dimensional BDD.

The size optimization presented in Section 5 can also be utilized as a second stage after the solution is decoded. Specifically, we consider the multiplied sequence of terminal labels as input and treat empty terminals as before. In order to respect the structure of a multi-dimensional BDD, we also need to have additional clauses guaranteeing that multiplied instances of the same original terminal have the same label. Note that the size encoding cannot be employed in a 1-stage approach since the structure of the BDD is not yet determined at first.

7 Experiments

In this section, we perform studies to experimentally analyze the performance of our tree (Section 4), size (Section 5), and multi-dimensional (Section 6) encodings for learning BDDs. We compare the performance of our approach against state-of-the-art BDD learning baseline and investigate the trade-off between size, accuracy, and performance in 1-stage, 2-stage, and multi-dimensional approaches to learning compact diagrams. Throughout the experiments, we seek to find out whether compactness can be achieved without significant compromise, and investigate its impact on testing accuracy. Furthermore, we aim to understand if the added expressiveness of multi-dimensional BDDs allows us to achieve high quality solutions with lower number of splits.

7.1 Setup

We use the Java programming language to produce encodings and the Loandra MaxSAT solver [4] to solve each instance. We set the solver timeout limit to 15 minutes and use the best found solution in case of a timeout. Our experiments are run on an AMD EPYC 7502 32-core processor and 256GB of RAM.

7.2 Baseline

We compare our approach against the recent work on SAT-based learning of BDDs. Hu et al. [16] aims to find BDD classifiers similar to our approach. However, unlike our approach, they only support binary classification and require pre-processing of each numeric feature into a set of binary features. Furthermore, they do not explicitly model the size of the reduced BDD.

Note that Hu et al. [16, 17] have compared optimal BDDs against optimal decision trees and heuristic decision trees and found that BDDs are competitive in terms of testing accuracy and have smaller size. Our approach aims to improve the performance of Hu et al. and learn more compact BDD classifiers with comparable accuracy. We therefore compare our approach to Hu et al. in terms of runtime, accuracy, and size of BDDs.

7.3 Datasets

We run our experiments over a range of datasets from the UCI repository [13] covering different number of labels, both numerical and binary features, and different dataset sizes.

7.4 Results

Results on Comparing Our Base Encoding Against Hu et al. [16]

In our first set of results, we compare the performance of our approach in terms of runtime and solution quality against Hu et al. for different numbers of splits. Note that while our approach learns multi-terminal BDDs, Hu et al.'s approach learns standard BDDs and is unable to support datasets with more than two labels. Furthermore, since the objective in Hu et al. does not include compactness considerations, we use our base encoding that only optimizes accuracy as well (Section 4). As both approaches explore the same space of (feasible and) optimal BDD solutions, we focus our comparison on optimization performance (i.e., training accuracy and runtime). In contrast, in the next two sets of experiments, we will also evaluate the testing accuracy over 5-fold cross-validation.

Based on the results presented in Table 1, we see that our approach is able to achieve a higher than or equal to Hu et al. [16] accuracy in all but one case. Furthermore, the runtimes of non-timeout cases show that our approach can also prove optimality much faster. As we expected given our direct encoding of non-binary values, the improvement is most noticeable in datasets with highly numerical features, namely Banknote and Ionosphere.

Results on 1-Stage and 2-Stage Size Optimization

Next, we evaluate the encoding presented in Section 5 to minimize the size of our learned BDDs. We perform size optimization in 1-stage and 2-stage approaches. In the 1-stage approach, we use different values for the weight of the size objective against the accuracy objective. Given a weight β and s_{max} splits, the combined size and accuracy objective aims to find a solution f with maximum $\beta(acc(f)) - (2^{s_{max}} - 1)|\mathcal{N}_D^f|$, where $\mathcal{N}_D^f$ is the set of decision nodes in the reduced version of f. We use $\beta = \infty^2$ to denote the 2-stage approach, $\beta = \infty^1$ to denote that accuracy is completely prioritized over size in the 1-stage approach, and Def. to denote the approach without any size optimization. Note that even in the 1-stage approach, running the second stage can further optimize the size if the first stage have yielded a sub-optimal solution (timeout). However, we opt against mixing the two approaches for the sake of clarity in comparison between the two. We use 5-fold cross validation and report the average value across folds to understand the effects of compactness on generalization and testing accuracy.

The results are presented in Figure 4. As expected, we see increase in training accuracy and size as we shift the priority to accuracy by changing the β value from 1 to ∞^2. However, the improvement in accuracy stagnates while the size continues to grow, indicating that a large enough β value can act as complete prioritization of accuracy. Interestingly, testing accuracy stagnates sooner and suffers from a larger variability as β increases, demonstrating a limited but positive effect of compactness on testing accuracy and generalization.

The solutions for lower β values in the 1-stage approaches are significantly smaller than the $\beta = \infty^2$ case, highlighting the importance of adding size considerations while the solution is being learned. However, optimizing size as a second stage still provides a significant size reduction compared to the case with no size optimization. For a detailed report of the results per dataset, we refer the reader to Table 2 in Appendix A.1.

Table 1 Results for learning max-accuracy BDDs.

Dataset			Splits	Accuracy (%)		Time (s)	
				Ours	Hu et al. [16]	Ours	Hu et al. [16]
Banknote			4	96.8	96.8	843.01	TO
$\|X\|$	$\|F\|$	$\|K\|$	5	97.6	90.4	TO	TO
1372	4	2	6	98.5	91.8	TO	TO
Breast			4	89.7	89.7	TO	TO
$\|X\|$	$\|F\|$	$\|K\|$	5	92.2	91.4	TO	TO
116	9	2	6	94.8	94.8	TO	TO
Cryotherapy			4	97.8	97.8	1.23	2.43
$\|X\|$	$\|F\|$	$\|K\|$	5	98.9	98.9	3.97	9.41
90	6	2	6	100	100	0.44	1.64
Immunotherapy			4	95.6	95.6	8.18	26.65
$\|X\|$	$\|F\|$	$\|K\|$	5	96.7	96.7	74.08	290.99
90	7	2	6	97.8	97.8	433.35	TO
Ionosphere			4	94.9	90.6	TO	TO
$\|X\|$	$\|F\|$	$\|K\|$	5	95.2	85.8	TO	TO
351	34	2	6	96.6	90.6	TO	TO
Iris			4	98.7	-	0.67	-
$\|X\|$	$\|F\|$	$\|K\|$	5	99.3	-	0.62	-
150	4	3	6	100	-	0.43	-
User			4	94.2	-	54.57	-
$\|X\|$	$\|F\|$	$\|K\|$	5	95.7	-	828.34	-
258	5	4	6	97.7	-	TO	-
Vertebral			4	88.1	87.7	TO	TO
$\|X\|$	$\|F\|$	$\|K\|$	5	89.7	90	TO	TO
310	6	2	6	91	90.3	TO	TO
Wine			4	99.4	-	29.29	-
$\|X\|$	$\|F\|$	$\|K\|$	5	100	-	2.07	-
178	13	3	6	100	-	1.14	-
Car			4	92.5	92.5	316.88	TO
$\|X\|$	$\|F\|$	$\|K\|$	5	92.9	92.9	TO	TO
1728	6	2	6	95.4	95.4	TO	TO
Monk2			4	74.6	74.6	85.49	473.43
$\|X\|$	$\|F\|$	$\|K\|$	5	84.6	84.6	185.3	416.37
169	6	2	6	100	100	0.35	1.09

Results on Learning Multi-Dimensional BDDs

In our next set of experiments, we evaluate our approach for directly learning multi-dimensional BDDs. Our goal is to understand whether the added expressiveness of multi-dimensional BDDs allows us to find high quality solutions with a lower number of splits. Furthermore, we aim to study the effects of dimension division for multi-dimensional BDDs

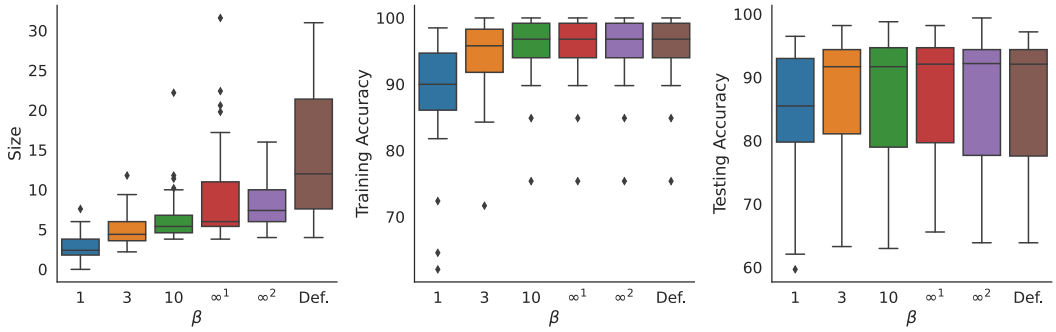


Figure 4 Distributional result of size, training accuracy, and testing accuracy across datasets for different β values.

by considering one balanced diagram with three multi-dimensional splits (2-2-2) and one unbalanced diagram with one 3-dimensional split followed by three 1-dimensional splits (3-1-1-1). Finally, we have used the size encoding in Section 5 to optimally decide the labels of empty terminals towards compactness in a second stage. Note that the size of a multi-dimensional BDD is considered to be its number of decision nodes ($|\mathcal{N}_D^f|$) after it is unfolded and reduced.

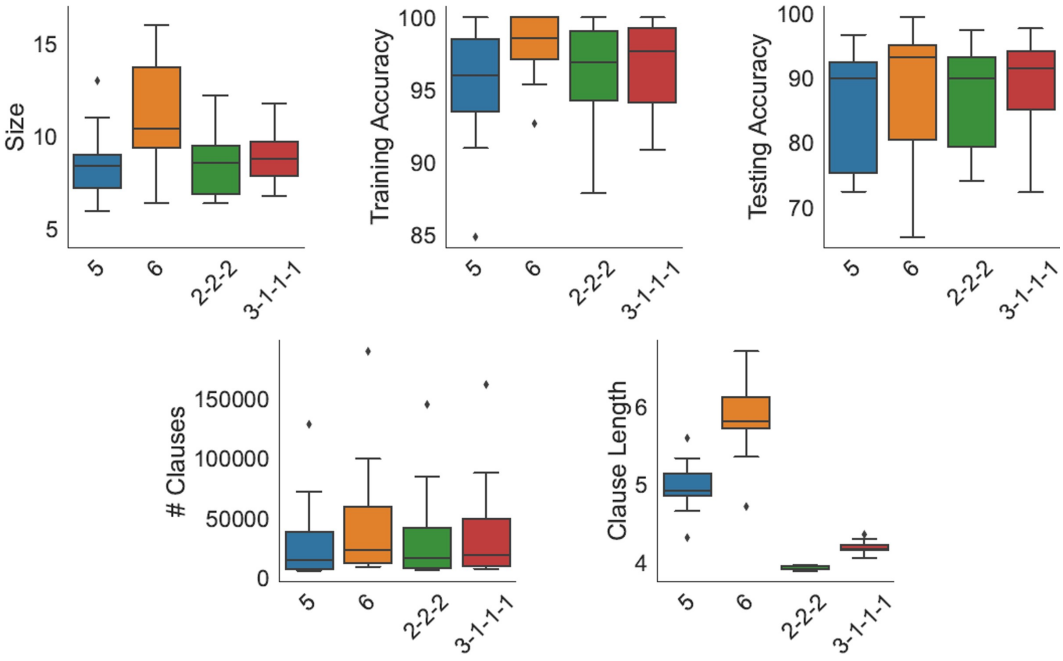


Figure 5 Distributional result of size, accuracy, number of clauses, and average clause length for BDDs of two number of splits and Multi-dimensional BDDs of two dimensions.

The results of the third set of experiments presented in Figure 5 show the two multi-dimensional approaches achieve training accuracy that is higher than a one-dimensional diagram with 5 splits and lower than a one-dimensional diagram with 6 splits. Since the sum of dimensions equal 6 both case, the upper bound is expected according to Theorem 11. However, the comparison against 5 splits, shows that the multi-dimensional approaches

perform better than the theoretical lower bound (resp. 3 and 4 splits) in practice. The same comparison is observed in testing accuracy and size. However, the 3-1-1-1 approach is able to achieve smaller variability in testing accuracy while still being close to the best approach in size. Finally, we see a lower number of clauses and a significantly shorter average clause length for our multi-dimensional approaches against ordinary BDDs of similar expressiveness. We refer the reader to Table 3 in Appendix A.1 for the complete results of this experiment with a larger set of dimension sequences considered.

Next, we run experiments for a significantly larger dataset, namely the Adult dataset ($|X| = 32561$, $|F| = 105$, $|K| = 2$), to investigate the difference in encoding size and performance between one-dimensional and multi-dimensional approaches on large datasets. Given the scale of the tasks, we increase the timeout limit to 60 minutes. We compare, one-dimensional BDDs against ones with 2, 3, and 4-dimensional splits according to their total number of dimensions in Figure 6 and Figure 7. Figure 7 depicts the exponential growth of encoding size for the one-dimensional approach. The training accuracy presented in Figure 6 shows that the exponential size causes the ordinary approach to decrease in quality and finally cease to produce any solutions due to memory overflow. We observe that by using two dimensional splits, we maintain a higher accuracy over a significantly larger number of dimensions. Three-dimensional and four-dimensional splits prove to be more challenging compared to two-dimensional splits. However, we are still able to find solutions for higher total number of dimensions compared to the one-dimensional splits. In this experiment, we avoid 5-fold cross-validation as we focus on the optimization performance of the two methods.

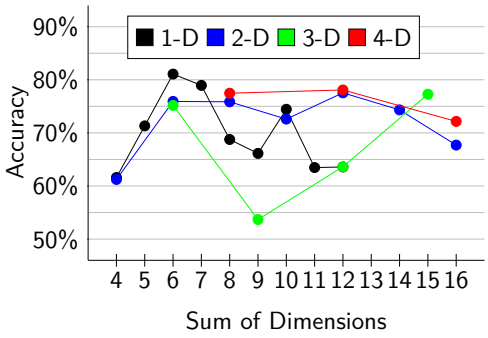


Figure 6 Training accuracy for the Adult dataset for different dimensions and number of total splits.

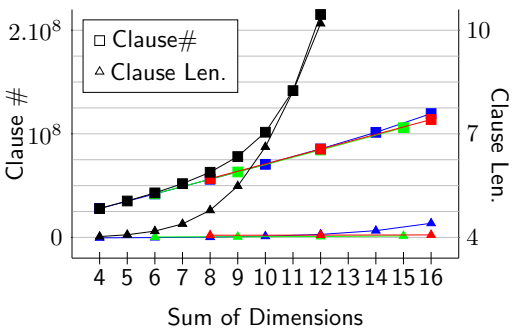


Figure 7 Number and average length of clauses of the encoding for the Adult dataset for different dimensions and number of total splits.

8 Conclusion

In this paper, we present a novel MaxSAT encoding for learning Binary Decision Diagrams. Our BDD encoding represents a tree which can be reduced to an equivalent diagram. We extend our encoding with optimization for the size of the reduced diagram. The size objective can be balanced against accuracy in a 1-stage approach or optimized as a second stage. Furthermore, we present a variant of our encoding using multi-dimensional splits, which are inner BDDs themselves. Our experiments show that we outperform the state-of-the-art SAT-based BDD learning baseline due to our direct encoding of numerical splits. We further show that our 1-stage and 2-stage size optimization approaches lead to significantly more compact solutions while maintaining testing accuracy. Finally, we show that the expressiveness of our multi-dimensional BDDs allows us to produce high quality solutions in smaller number of splits, mitigating the exponential growth in the size of the encoding.

Our work, can be extended in a number of ways. Our size encoding for the 2-stage optimization can be extended by allowing more than empty terminal labels to be changed, e.g., feature ordering or splits. Moreover, nested directional BDDs can be added to our multi-dimensional BDD encoding to improve expressiveness and avoid exponentiation even further. Other interesting directions for future work involve investigating different strategies for balancing the two objectives, e.g., producing Pareto optimal solutions, or a more comprehensive analysis of parameters including but not limited to the dimension sequences. Finally, investigating the impact of more expressive BDDs such as free BDDs, which are not constrained to be ordered [32], is also an interesting direction for future research.

References

1 Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. Learning optimal decision trees using caching branch-and-bound search. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 3146–3153, 2020.

2 Sheldon B. Akers. Binary decision diagrams. *IEEE Transactions on computers*, 27(06):509–516, 1978.

3 Florent Avellaneda. Efficient inference of optimal decision trees. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 3195–3202, 2020.

4 Jeremias Berg, Emir Demirović, and Peter J Stuckey. Core-boosted linear search for incomplete MaxSAT. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pages 39–56. Springer, 2019.

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

6 Allan Borodin, Alexander Razborov, and Roman Smolensky. On lower bounds for read-k-times branching programs. *Computational Complexity*, 3:1–18, 1993.

7 Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

8 Randal E Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.

9 Gianpiero Cabodi, Paolo E Camurati, Alexey Ignatiev, Joao Marques-Silva, Marco Palena, and Paolo Pasini. Optimizing binary decision diagrams for interpretable machine learning classification. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1122–1125. IEEE, 2021.

10 Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

11 Edmund M Clarke, Masahiro Fujita, and Xudong Zhao. Multi-terminal binary decision diagrams and hybrid decision diagrams. *Representations of discrete functions*, pages 93–108, 1996.

12 Adnan Darwiche. Sdd: A new canonical representation of propositional knowledge bases. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

13 Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL: `http://archive.ics.uci.edu/ml`.

14 Alexandre M Florio, Pedro Martins, Maximilian Schiffer, Thiago Serra, and Thibaut Vidal. Optimal decision diagrams for classification. *arXiv preprint arXiv:2205.14500*, 2022.

15 Oktay Günlük, Jayant Kalagnanam, Minhan Li, Matt Menickelly, and Katya Scheinberg. Optimal decision trees for categorical data via integer programming. *Journal of Global Optimization*, pages 1–28, 2021.

16 Hao Hu, Marie-José Huguet, and Mohamed Siala. Optimizing binary decision diagrams with maxsat for classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36(4), pages 3767–3775, 2022.

17 Hao Hu, Marie-José Huguet, and Mohamed Siala. Optimizing binary decision diagrams with maxsat for classification, 2022. `arXiv:2203.11386`.

18 Hao Hu, Mohamed Siala, Emmanuel Hébrard, and Marie-José Huguet. Learning optimal decision trees with MaxSAT and its integration in AdaBoost. In *International Joint Conference on Artificial Intelligence and Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI)*, 2020.

19 Alexey Ignatiev and Joao Marques-Silva. SAT-based rigorous explanations for decision lists. In *Theory and Applications of Satisfiability Testing–SAT 2021: 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings 24*, pages 251–269. Springer, 2021.

20 Alexey Ignatiev, Filipe Pereira, Nina Narodytska, and Joao Marques-Silva. A SAT-based approach to learn explainable decision sets. In *Automated Reasoning: 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings 9*, pages 627–645. Springer, 2018.

21 Dmitry Ignatov and Andrey Ignatov. Decision stream: Cultivating deep decision trees. In *2017 ieee 29th international conference on tools with artificial intelligence (ictai)*, pages 905–912. IEEE, 2017.

22 Stasys Jukna. A note on read-times branching programs. *RAIRO-Theoretical Informatics and Applications*, 29(1):75–83, 1995.

23 Donald E Knuth. *The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams*. Addison-Wesley Professional, 2009.

24 Ron Kohavi. Bottom-up induction of oblivious read-once decision graphs. In *Machine Learning: ECML-94: European Conference on Machine Learning Catania, Italy, April 6–8, 1994 Proceedings 7*, pages 154–169. Springer, 1994.

25 Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model, 2015.

26 Breiman LI, Jerome Friedman, RA Olshen, and C.J. Stone. Classification and regression trees (CART). *Biometrics*, 40:358, September 1984. `doi:10.2307/2530946`.

27 Joao Marques-Silva and Alexey Ignatiev. Delivering trustworthy AI through formal XAI. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36(11), pages 12342–12350, 2022.

28 Bernard ME Moret. Decision trees and diagrams. *ACM Computing Surveys (CSUR)*, 14(4):593–623, 1982.

29 J Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

30 J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

31 Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. nat mach intell 1: 206–215. *DOI: https://doi.org/10.1038/s42256-019-0048-x*, 2019.

32 Petr Savickỳ and Ingo Wegener. Efficient algorithms for the transformation between different types of binary decision diagrams. *Acta Informatica*, 34(4):245–256, 1997.

33 Pouya Shati, Eldan Cohen, and Sheila McIlraith. SAT-based approach for learning optimal decision trees with non-binary features. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

34 Pouya Shati, Eldan Cohen, and Sheila A. McIlraith. SAT-based optimal classification trees for non-binary data. *Constraints*, July 2023. `doi:10.1007/s10601-023-09348-1`.

35 Jamie Shotton, Toby Sharp, Pushmeet Kohli, Sebastian Nowozin, John Winn, and Antonio Criminisi. Decision jungles: Compact and rich models for classification. *Advances in neural information processing systems*, 26, 2013.

36 Hazem Torfah, Shetal Shah, Supratik Chakraborty, S Akshay, and Sanjit A Seshia. Synthesizing pareto-optimal interpretations for black-box models. In *2021 Formal Methods in Computer Aided Design (FMCAD)*, pages 153–162. IEEE, 2021.

37 Hélene Verhaeghe, Siegfried Nijssen, Gilles Pesant, Claude-Guy Quimper, and Pierre Schaus. Learning optimal decision trees using constraint programming. *Constraints*, 25(3):226–250, 2020.

38 Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021.

A Appendix

A.1 Additional Experimental Results

Table 2 Average 5-fold results for learning BDDs with the combined objective of accuracy and compactness.

D.S.	Splits	Size					Training Acc. (%)					Testing Acc. (%)				
	β	1	3	10	∞^1	∞^2	1	3	10	∞^1	∞^2	1	3	10	∞^1	∞^2
Banknote	4	1	3.6	4	4	4.4	85.3	95.8	96.8	96.8	96.8	85.2	95	96.6	96.6	96.4
	5	3	4	5	11	6	94	96.8	97.6	97.5	97.6	93.7	96.4	96.9	96.8	96.7
	6	4	6	5.8	19.8	6.4	96.8	98.6	98.4	98.6	98.6	96.5	97.4	97.4	97.2	97.2
Breast	4	1	4	5.2	6.4	6.8	72.4	88.8	90.7	90.7	90.7	71.5	75.9	75.9	78.4	77.6
	5	3.8	5.4	11.4	11.8	11	88.8	91.8	94	94	94	74.1	70.7	74.1	72.4	72.5
	6	5.6	9.4	22.2	22.4	16	92.9	96.1	97	97.2	97.2	75	71.6	63	71.6	65.5
Cryo.	4	1	3	4.8	4.8	6	86.1	94.7	97.8	97.8	97.8	75.6	88.9	92.2	92.2	94.4
	5	2.4	5.2	5.8	5.8	8	93.1	99.2	99.4	99.4	99.4	83.3	94.4	91.1	91.1	90
	6	4.6	5.8	5.8	5.8	10.4	98.3	100	100	100	100	95.6	87.8	90	90	82.2
Immuno.	4	1	2.2	5.4	5.4	6.8	87.2	91.4	95.6	95.6	95.6	82.2	83.3	83.3	83.3	85.6
	5	1.6	4.4	5.4	5.4	8.4	89.7	95.3	96.7	96.7	96.7	81.1	81.1	78.9	81.1	73.3
	6	3.8	7.2	10	10	14	94.7	98.3	99.2	99.2	99.2	80	76.7	73.3	72.2	75.6
Iono.	4	1.8	2.4	4	5.6	4	90	92	95.2	94.9	95.2	87.2	90.9	90.9	90.3	92.3
	5	2	3.6	5.2	10.2	8.4	91.2	94.4	95.9	95.7	95.9	89.7	91.7	89.5	85.8	89.2
	6	2.4	6	7	17.2	12.8	92	96.3	96.9	97.2	97.1	90.9	88.6	88	86.9	86.3
Iris	4	2	2.4	3.8	3.8	4.2	96.3	97.3	98.8	98.8	98.8	94	94	94.7	94.7	94
	5	2	2.8	5.2	5.2	6	96.3	98	99.5	99.5	99.5	94	95.3	96.7	96	96
	6	2.8	5.4	5.8	5.8	7	98	99.8	100	100	100	95.3	95.3	94.7	96	94.7
User	4	2.2	4.4	6	6	6	82.8	91.5	94.3	94.3	94.3	79.8	87.6	93.4	93.4	93.4
	5	4	6	6.2	9.2	7	90.5	95.9	96	96	96	86	93	91.9	93.8	92.2
	6	6	7	8	31.6	9.2	96.2	97.2	97.8	97.8	97.8	93	93	95.7	95.7	95.4
Vertebral	4	1.2	2.4	4	5.4	4.6	81.8	86.8	89.8	89.8	89.8	76.8	76.5	79	78.1	77.7
	5	2.2	4.2	5.2	10.6	8.6	86.1	89.4	89.8	90.8	91	74.8	81.6	81.9	79.7	76.1
	6	3.4	6.2	10.2	20.6	15	88.5	91.6	92.4	92.8	92.7	80.3	80.3	79	76.8	78.7
Wine	4	2.2	3	4	5	6.4	93.8	98.5	99.6	99.9	99.9	87.7	96.6	93.3	93.3	93.9
	5	3	3.6	4.6	4.6	9.4	98.5	99.3	100	100	100	95	95	96.7	95.5	92.7
	6	3	4.6	4.6	4.6	10	98.5	100	100	100	100	95	93.3	92.2	93.3	93.3
Car	4	2	4	4	4	5.4	85.5	92.5	92.5	92.5	92.5	85.5	92.5	92.5	92.5	92.5
	5	3.2	4	4.6	6.8	7.4	90	92.5	92.9	93.1	93.1	88.6	92.5	91.7	92.1	92.1
	6	4	6.6	6.8	12.2	9.6	89	95	95	95.4	95.4	87.6	93.8	93.9	94.7	94.7
Monk2	4	0	3.4	5.4	5.4	6.6	62.1	71.7	75.4	75.4	75.4	62.1	63.3	64.5	65.6	63.9
	5	0.6	7.8	9	9	13	64.6	84.3	84.9	84.9	84.9	59.7	79.3	76.4	77.6	74.6
	6	7.6	11.8	11.8	11.8	13.4	86.2	100	100	100	100	81.7	98.2	98.8	98.2	99.4

Table 3 Results for learning max-accuracy multi-dimensional BDDs.

Dataset	Dimensions	Accuracy (%)		Size		Time (s)
		Training	Testing	Stage 1	Stage 2	
Banknote	2-2-2	**98.6**	97.5	7.4	**6.6**	TO
	3-3	98.4	97.5	**6.6**	**6.6**	TO
	1-2-3	**98.6**	96.9	8.2	6.8	TO
	3-1-1-1	**98.6**	**97.7**	8.8	6.8	TO
Breast	2-2-2	**94.6**	**74.1**	**8.6**	**8.6**	TO
	3-3	92.9	64.6	8.8	8.8	TO
	1-2-3	94.2	68.1	9.6	9.6	TO
	3-1-1-1	94.2	72.4	11	9.8	TO
Cryotherapy	2-2-2	99.7	86.7	10.6	10.2	**6.18**
	3-3	99.4	**92.2**	10.6	10	25.29
	1-2-3	99.7	90	10.6	9.8	13.66
	3-1-1-1	**100**	86.7	**10**	**9.4**	6.45
Immunotherapy	2-2-2	97.2	75.6	**9.2**	**8.6**	515.83
	3-3	96.9	76.7	10.2	10.2	750.07
	1-2-3	97.2	76.7	11.6	11.2	767.69
	3-1-1-1	**98.1**	**84.4**	9.8	9.6	**489.51**
Ionosphere	2-2-2	96.9	90	**6.6**	**6.6**	TO
	3-3	95.4	88.3	8.8	8.8	TO
	1-2-3	**97**	90.3	7.4	7.4	TO
	3-1-1-1	96.7	**91.5**	8.2	8.2	TO
Iris	2-2-2	99.5	**95.3**	**10.4**	8.4	1.17
	3-3	99.5	94.7	11.2	11.2	0.67
	1-2-3	99.8	94	14.6	**8**	1.2
	3-1-1-1	**100**	93.3	12.4	8.8	**0.44**
User	2-2-2	95.6	91.1	**12**	11.6	TO
	3-3	93	88.7	12.2	12.2	**143.96**
	1-2-3	**97.7**	**96.1**	18.2	18.2	494.02
	3-1-1-1	**97.7**	95.4	13.6	**9.8**	783.39
Vertebral	2-2-2	91.3	79.4	7.6	**7.2**	TO
	3-3	**91.8**	**80**	**7.4**	7.4	TO
	1-2-3	91.1	79.4	9.4	8.2	TO
	3-1-1-1	90.9	77.4	10.6	8.6	TO
Wine	2-2-2	**100**	93.9	**9**	8.8	3.43
	3-3	**100**	90.5	9.6	9.6	**1.03**
	1-2-3	**100**	93.3	14.2	11.6	5.29
	3-1-1-1	**100**	**95**	**9**	**7.6**	9.03
Car	2-2-2	94	92.5	6.4	6.4	TO
	3-3	93	92.3	6.4	6.4	TO
	1-2-3	92.8	92.6	**6.2**	**5.8**	TO
	3-1-1-1	**94.1**	**93.2**	6.8	6.8	TO
Monk2	2-2-2	87.9	79.3	12.2	12.2	TO
	3-3	89.6	80.5	**10.6**	**10.6**	TO
	1-2-3	90.1	74.6	13	13	TO
	3-1-1-1	**92.9**	**85.8**	11.8	11.8	**695.58**