DOCTOR OF SCIENCES

Constraint Enforcement on Decision Trees and its Application to Global Explainability

Nanfack, Geraldin

Award date:
2022

Awarding institution:
University of Namur

Link to publication

Constraint Enforcement on Decision Trees and its Application to Global Explainability

Géraldin Nanfack

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Science in Computer Science

December 14, 2022

Contents

Abstract

Machine learning is taking an increasing place in our society, even in high-stake and highly regulated domains such as finance and health. In such domains, automatic decisions produced by machine learning models may be expected to meet guidelines. However, as a research field, machine learning has been much more focused on pushing the boundary of performance of existing models, or on designing more powerful models. As a result, less attention has been paid to either the enforcement of guidelines or the incorporation of domain knowledge to prevent undesirable behaviour.

This thesis first focuses on constraint enforcement on one of the simplest, nonlinear and most popular classes of machine learning models which are decision trees. The thesis also uses constraint enforcement to improve the explainability by decision rules of differentiable black-box models. The thesis makes three contributions to machine learning research.

First, the thesis thoroughly investigates and analyses techniques available in the literature that tackle the imposition of constraints on decision trees. In particular, the thesis proposes a taxonomy of constraints and another taxonomy of approaches through the lens of optimisation tools. The taxonomy of constraints defined structure-level constraints (e.g., size, depth), feature-level constraints (e.g., monotonicity, fairness), instance-level constraints (e.g., robustness). The taxonomy of methods includes top-down greedy, safe enumeration, LP/SAT/CP and probabilistic (including Bayesian) approaches.

Second, still considering decision tree models, the thesis introduces two techniques to enforce constraints on decision trees. The first technique leverages soft constraint enforcement and introduces a method called BDT, which integrates boundary-based fairness constraints to learn fairer decision trees. The second technique called CPTree leverages hard constraint enforcement and introduces a framework based on MIP/CP to learn decision trees under domain-knowledge constraints.

Third, considering differentiable black-box models such as multi-layer perceptrons (MLPs) the thesis introduces a statistical framework where these models can be implicitly and softly constrained to be easily explainable by decision rules.

Acknowledgement

“Failure is an incredible learning experience. It teaches you humility. It teaches you to work harder. It is the first step to understanding”.
Richard Feynman

*** The following text is intentionally written in French language. ***

En tant que chrétien catholique, mon ultime remerciement va à l’endroit de Dieu, en qui sa Grâce, non pas la chance ni mes compétences, a amplement suffit pour rencontrer des personnes merveilleuses qui m’ont aidé à accomplir ce travail. Merci Seigneur!

Après avoir dit cela, je tiens à remercier les institutions qui m’ont permis d’effectuer ce travail. Je remercie alors l’Université de Namur qui m’a accueillie pendant ces quatre années. Je remercie également le FNRS et le projet EOS VeriLearn qui, grâce au financement, m’ont permis de m’éloigner des besoins primaires pour me consacrer sur ma thèse. Merci! Je remercie très subtilement, l’école Polytechnique de Yaoundé où j’ai passé mes trois premières années universitaires et l’École Nationale Supérieure d’Analyse des Systèmes de Rabat qui m’a permis d’obtenir mon Master. Merci!

La toute première personne à qui j’exprime ma profonde gratitude est naturellement Benoît, mon promoteur. Merci Benoît de m’avoir sélectionné comme doctorant pour ce projet de thèse. J’avais choisi de faire ma thèse avec toi pour te ressembler dans ta rigueur scientifique, ta bravoure et la qualité impressionnante de tes travaux. De façon absolue, je n’ai appris qu’une toute petite fraction de compétences d’un chercheur scientifique. Mais de façon relative, par rapport à ce que je disposais avant de commencer cette thèse, j’ai énormément appris grâce au travail avec toi. J’ai connu pas mal d’échecs, entièrement propres à moi. Ma passion pour la science est restée toujours inébranlable parce que malgré tout, le magnifique humain a toujours lancé un appel pour me soutenir. Merci Benoît!

Je vais aussi remercier infiniment mon co-promoteur Paul. J’ai probablement été très embarassant dans mon caractère, mon anglais, mes erreurs. Tu as toujours su la formule adéquate pour me le faire remarquer. Ton sens critique et ta curiosité scientifique m'ont beaucoup aidé à m'améliorer, à travailler davantage. Intrinsèquement régulièrement têtu, j'ai appris à être humble si bien que j'ai quasiment toujours tendance à venir t'embêter pour demander conseils. Merci Paul!

Je remercierai aussi mon jury de thèse. Merci d'abord aux membres de mon comité d'accompagnement Michel et Gilles pour tout le feedback donné à mes travaux! Merci ensuite au président du jury Wim et un grand merci à Hendrik pour toutes ses questions et suggestions qui m'ont significativement aidé à être précis et à identifier mes faiblesses.

Je remercierai brièvement tous mes collègues à l'UNamur. Je suis désolé de ne pas tous vous citer. Je citerai ici mes collègues/collaborateurs du bureau avec qui nous avons eu beaucoup d'échanges scientifiques. Adrien, Minh, Valentin, Becca, Jérôme, Charline, Pierre, merci à vous! Merci aussi particulièrement aux amis du mini-foot du mercredi, en particulier Pol, Loup et Mathieu!

Je remercierai énormément ma mère, elle qui me disait durant mon enfance qu'observant mes résultats, je peux tout faire. Grâce à elle, j'ai connu la "niaque", qui m'aide durablement à vaincre tous mes obstacles. Merci Mater! Un grand merci également est rendu à mon papa, qui, souvent contre son gré, a accepté mes choix comme celui d'opter pour le génie informatique alors qu'il aurait surtout voulu que son fils fasse le génie civil. Merci Pater! Merci à tous mes frères et soeurs qui m'ont toujours soutenu. Merci Judicaëlle, Kévin, William, Juvenale, Aurel et Brayan! J'ai une large famille... je ne saurai malheureusement pas tous vous citer. Mais un clin d'oeil à Gauss, Ange, Idriss, Paterne et le Père Josué. Merci famille et amis très particuliers!

Je remercierai enfin Danielle, ma fiancée. Depuis 2015, tu as toujours été à mes côtés, m'a soutenu à distance quand j'étais en séjour au Maroc et durant les deux premières années de ma thèse. Tu as fait beaucoup de compromis de ta vie pour être avec moi. Malheureusement,je n'en ai pas fait beaucoup. Merci Princesse!

Sans tous vous nommer, tous mes amis, toutes mes amies, personnes rencontrées, je n'aurai pas beaucoup d'occasions de vous le dire. Merci!

Chapter 1

Introduction

This thesis is entitled "Constraint Enforcement on Decision Trees and its Application to Global Explainability". It propses techniques to infuse domain knowledge into specific supervised machine learning (ML) models. These techniques are mainly approached from a constrained optimisation perspective. Therefore, the thesis lies at the intersection between supervised machine learning and constrained optimisation.

1.1 Context, Scope and Motivation

Almost everywhere where automatic decisions can be made, machine learning is taking an increasing place in our society. Examples range from applications in finance, human resource management (job hiring), health (computer-aided diagnosis) to chemistry (computer-aided drug design) (Mehrabi *et al.*, 2021; Ren *et al.*, 2022). This increasing place taken by machine learning stems from the ease of data collection, which is partially caused by the growth of computer storage and power, and also by the increasingly digital world. However, this place intersects with several spheres of our society, which are highly regulated. Hence, automatic decisions produced by machine learning models are expected to meet guidelines. Otherwise, these models could violate laws by producing, for example, illegal decisions.

As a research field, machine learning has been much more focused on pushing the boundary of performance of existing models, or on designing more powerful models. As a result, less attention has been paid to either the enforcement of guidelines or the incorporation of domain knowledge to prevent undesirable behaviour. Moreover, even when guidelines (usually in the form of prior knowledge) are incorporated into models, these, instead, have been made to improve performance (e.g., in the case the number of data instances is small) rather than to seek certain guarantees on produced models.

As a concrete example, consider an artificial intelligence (AI) system in a bank, which leverages an ML model whose task is to predict whether a customer should be granted credit. This AI system and the predictive ML model, operating in an ecosystem highly regulated by laws and domain-related best practices, are likely to meet stringent requirements. For example, besides being accurate as much as possible, it may be desirable that the ML model should not explicitly as well as implicitly use some sensitive features such as gender to make predictions. Indeed, removing the sensitive feature may not solve the issue because a sensitive feature such as the race may be highly correlated with other features such as district or quarter of residence. In addition, this ML model may also be required to give reasons in human-understandable terms why it produced some predictions. The model may also be required to not disclose the identity of customers so that the bank can use this guarantee for marketing strategies.

In this thesis, constraints on the model will represent the above discussed guidelines or requirements. As there are several constraints that can depend on the type of models, and since there are plenty of models, this thesis is focused on two supervised ML models: decision tree models because of their relative simplicity and multi-layer perceptions because they are the basis of the preeminent sub-field of ML, which is deep learning. However, some of the techniques presented can be applied to other supervised models. Additionally, this thesis mainly focuses on tabular data, though again, some of the techniques developed can be applied to different data modalities. The overall goal is to learn models that could guarantee a satisfaction of constraints.

1.2 Research Problems

In order to achieve the goal of this thesis that aims to develop techniques to easily integrate constraints on decision trees and MLP models, we formulate three research questions. The constraints considered are domain-knowledge constraints (fairness constraints included) for decision trees and the explainability constraints on the black-box MLPs. This thesis is therefore guided by the following research questions.

1. **RQ1:** How does the literature tackle the problem of learning decision tree

models under constraints? More specifically, (**RQ1.1**) What are the constraints applied to decision trees? And (**RQ1.2**) what are the corresponding methods to learn constrained decision trees? The goal of this research question is to have a thorough analysis of related works. This analysis highlighted several gaps in the literature, whose two of them are expressed in the next research questions.

2. **RQ2:** How to learn decision trees under domain-knowledge constraints? This research question will be viewed in the lens of constrained optimisation.

3. **RQ3:** How to leverage constraint enforcement to learn more easy-to-explain (globally) black-box models such MLPs? According to the related work studied thanks to **RQ1**, decision trees are well used to globally explain "black-box" models. However, very little to no work examine how compatible (or easy to explain by a decision tree) a black-box is. **RQ3** aims to leverage constraint enforcement to integrate this requirement for a black-box model.

1.3 Contributions

The previous research questions correspond to the contributions of this thesis. These contributions are presented in details in the chapters.

In particular, first, Chapter 3 aims to answer **RQ1** by studying and analysing the related works with the emphasis on optimisation. Second, Chapter 4 proposes a technique to impose fairness constraints on decision trees. Third, Chapter 5 introduces a flexible framework to learn decision trees under domain-knowledge constraints. Both chapters 4 and 5 aim to answer **RQ2**. Four, Chapter 6 answers the research question **RQ3** by presenting a framework where differentiable black-box models can be constrained to be more easily and globally explainable by decision rules. The global discussion is presented in Chapter 7. Chapter 8 finally presents the conclusion and future works.

The following works were produced during my own

1. **Nanfack, Géraldin**, Delchevalerie, Valentin, and Frénay, Benoît. "Boundary-Based Fairness Constraints in Decision Trees and Random Forests". In Proc. of the 29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. 2021. p. 375-380;

2. **Nanfack, Géraldin**, Temple, Paul, and Frénay, Benoît. "Global explanations with decision rules: a co-learning approach". In Proc. of the 37th Conference on Uncertainty in Artificial Intelligence. PMLR, 2021. p. 589-599.;

3. **Nanfack, Géraldin**, Temple, Paul, and Frénay Benoît. “Constraint Enforcement on Decision Trees: a Survey”. ACM Computing Surveys (CSUR). 2022;

4. **Nanfack, Géraldin**, Temple, Paul and Frénay, Benoît. “Learning Customised Decision Trees under Domain-knowledge Constraints”. Under revision in the Pattern Recognition journal. 2022.

Apart from the above scientific productions, other works have been done through collaborations with colleagues. These works are not closely related to my thesis. They are listed bellow.

1. Bibal, Adrien, VU, Viet Minh, **Nanfack, Géraldin**, and Frénay, Benoît. “Explaining t-SNE embeddings locally by adapting LIME”. In Proc. of the 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. 2020. p. 393-398.

2. Stassin, Sédrick, **Nanfack, Géraldin**, Englebert, Alexandre, Peiffer, Gilles, Albert, Julien, Versbraegen, Nassim, Doh, Miriam, Riche, Nicolas, Frénay, Benoît, De Vleeschouwer, Christophe. “An Experimental Investigation into the Evaluation of Explainability Methods”. 2022. To be submitted to PAKDD’23.

3. Blockeel, Hendrik, Devos, Laurens, Frénay, Benoît, **Nanfack, Géraldin**, Nijssen, Siegfried. “Decision Tree: Past, Present and Future”. 2022. To be submitted to the journal Frontiers in Artificial Intelligence.

Chapter 2

Background

Contents

This chapter introduces supervised machine learning with a mix of inspiration from Shalev-Shwartz and Ben-David (2014) and Murphy (2012). The chapter also presents the class of models that are used in this thesis and defines domain-constraints that will be used.

2.1 Supervised Machine Learning

In machine learning, we have two main quantities of interest. We consider a sample of observations $\mathcal{D}$ and an (unknown) quantity $\boldsymbol{\theta}^*$ that represents a function of a system that we want to learn. There exists two learning approaches depending on how we see the learning problem: the frequentist and the Bayesian. While the frequentist approach tends to focus more on the sampling properties (in terms of $p(\mathcal{D}|\boldsymbol{\theta})$ [1]) of estimators of $\boldsymbol{\theta}^*$, the Bayesian approach focuses more on the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. In the following, we describe the frequentist approach of supervised machine learning, in particular for classification tasks, which is the most common setting. The description can be easily extended for regression tasks.

In supervised ML, given a classification task with a data sample $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where the inputs-output pairs $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$, the goal is to learn the mapping f[2]: $\mathcal{X} \longrightarrow \mathcal{Y}$. Datapoints or instances $\boldsymbol{x}_i$ are vectors of observations of M random variables $X_1, X_2, ..., X_m$. They are usually assumed to be independently, indentically and distributed (i.i.d.) according to a probability distribution p_{data} over $\mathcal{X}$ before being labelled by the function f, i.e., $y_i = f(\boldsymbol{x}_i)$.

In practice the *true* function f as well as the *true* data distribution p_{data} that allowed to generate data are usually unknown. We typically assume a hypothesis class $\mathcal{H} = \{f_{\boldsymbol{\theta}}, \boldsymbol{\theta} \in \Theta\}$, powerful enough to hopefully contain f. The goal is to learn an hypothesis $f_{\hat{\boldsymbol{\theta}}}$ that minimises the error:

$$\mathcal{L}_{(p_{\text{data}}, f)}(f_{\hat{\boldsymbol{\theta}}}) = \mathbb{P}_{\boldsymbol{x} \sim p_{\text{data}}} \left[f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}) \neq f(\boldsymbol{x}) \right]. \tag{2.1}$$

$\mathcal{L}_{(p_{\text{data}}, f)}(f_{\hat{\boldsymbol{\theta}}})$ is called the *generalisation error* or the *risk* or the *true error* of $f_{\hat{\boldsymbol{\theta}}}$.

2.1.1 Model Fitting

Again, despite being unknown, p_{data} appears on the right hand side of Eq.2.1. We therefore eventually use the empirical distribution by relying on the (training) sample $\mathcal{D}$ to learn the function $f_{\hat{\boldsymbol{\theta}}}$ through the empirical risk minimisation

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} L\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i\right), \tag{2.2}$$

[1] $\boldsymbol{\theta}$ is an estimation of $\boldsymbol{\theta}^*$

[2] f here is similar to $\boldsymbol{\theta}^*$.

where $L : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}_+$ (or simply $\mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}_+$) is a loss function. Solving the optimisation problem of Eq. 2.2 is called *model fitting*.

We will mainly use two classical losses in this thesis despite there are plenty that exist.

1. The $0-1$ loss: it is expressed as follow

$$L_{0-1}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i\right) = \begin{cases} 1 & \text{if} \quad f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \neq y_i, \\ 0 & \quad \text{Otherwise.} \end{cases}$$

2. The negative log likelihood: when $f_{\boldsymbol{\theta}}$ can be viewed as a (conditional) probability distribution over $\mathcal{X} \times \mathcal{Y}$, e.g, simply $p(\mathrm{y}|\mathbf{x};\boldsymbol{\theta})$, one classical loss is the negative log of this probability. This can be expressed as

$$L\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i\right) = -\log p(y_i|\boldsymbol{x}_i;\boldsymbol{\theta}).$$

   Minimising Eq. 2.2 using this loss[3] gives the maximum likelihood estimate (MLE) $\hat{\boldsymbol{\theta}}_{\text{MLE}}$, which rewritten as

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\min_{\boldsymbol{\theta}} -\sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i;\boldsymbol{\theta}), \tag{2.3}$$

2.1.2 Model Evaluation

After choosing the hypothesis space $\mathcal{H}$ and fitting the model with Eq. 2.2, one must evaluate the ability of the model to generalise on unseen data. This is done by estimating the true risk in Eq. 2.1 using additional set of examples, independent from the training sample $\mathcal{D}$, but assumed to be sampled according to p_{data}.

In several situations, we split the available examples in three sets: training, validation and test sets. We use the first set to learn or to fit models. The second set, which is the validation set is used for model selection. The last set, the test set is used to estimate the true error of the model.

2.2 Decision Trees and Multi-Layer Perceptrons

There exist several hypothesis classes $\mathcal{H}$. This section describes two particular classes of supervised learning models.

[3]Note the absence of the term $\frac{1}{N}$, which has no effect on the optimisation problem.

2.2.1 Decision Trees

A decision tree is a class of machine learning models primarily introduced in the supervised setting, although they have also been extended to unsupervised learning tasks such as clustering (Blockeel *et al.*, 1998). The following describes the classification trees, inspired from Murphy (2022, 2012). Regression trees are easily extendable with this description.

A decision tree (e.g, see Fig. 2.1) can be described using the graph formalism of Safavian and Landgrebe (1991). Using this formalism, a decision tree can be viewed as a directed acyclic graph, with one node (called the root node) without incoming edges, where (i) every node except the root node has one incoming edge and (ii) there is a unique path from each node to the root node. Each node that has at least two child nodes is called *internal* nodes and nodes that do not have child nodes are called leaf nodes. A tree is said to be *binary* if each internal node (the root included) has exactly two child nodes. Each edge on the tree is labelled by a splitting rule (e.g., $X_1 \leq \alpha_{11}$) that has been produced by the learning algorithm. A splitting rule can involve only one variable (as in Fig. 2.1): in this case, the decision tree is said to be *univariate.* Finally, each leaf node is labelled by a class. It is worth mentioning that a decision tree geometrically defines a partitioning of the input space $\mathcal{X}$ into a set of disjoint regions $R_k, k = 1..L$, where L is the number of leaf nodes.

The decision rule function i.e., the mapping from the input space to the output space of the decision tree can be expressed as $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{k=1}^{L} w_k \mathbb{1}_{R_k}(\boldsymbol{x})$, where $\mathbb{1}_{\mathbb{A}}$ is the indicator function over the set $\mathbb{A}$ and w_j is the label of the k-th leaf node or the region R_k. If R_k is viewed as $R_k = \{\alpha_{kd}^{(1)} \leq x_d \leq \alpha_{kd}^{(2)}\}_{d=1}^{M}$, where $\alpha_{kd}^{(i)} \in \bar{\mathbb{R}}$ are the boundaries[4] and $\boldsymbol{\theta} = \{(R_k, w_k), k = 1..L\}$, the decision tree learning problem can be framed as a minimisation problem of the following loss

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{k=1}^{L} \sum_{\boldsymbol{x}_i \in R_k, i=1..N} L_{0-1}(y_i, w_k). \tag{2.4}$$

As all the R_k share some parameters depending on the tree structure and as they are involved in $\boldsymbol{\theta}$, one can see that the loss in Eq.2.4 therefore depends on the discrete tree structure. By analysing the search space of the partitioning class, Das and Goodrich (1997) (resp. Hyafil and Rivest (1976)) show that finding the optimal decision tree that minimises Eq. 2.4 (resp. the size of tree) is NP-complete.

[4]Note that the upper boundary $\alpha_{kd}^{(2)}$ can be $+\infty$ and the lower boundary $\alpha_{kd}^{(1)}$ can be $-\infty$, whenever relevant.

In practice, one of the well-known method to learn decision trees is a top-down greedy procedure in which we recursively (or iteratively) grow the tree by adding one node at a time. This is the approach used by the popular CART (Breiman *et al.*, 1984), C4.5 (Quinlan, 1993) and ID3 (Quinlan, 1986) algorithms.

Algorithm 2.1 Recursive procedure to learn a classification binary tree

```
Input: training set $\{\mathbf{x}_i, y_i\}_{i=1}^N$
Output: A decision tree $f_{\boldsymbol{\theta}}$, with $\boldsymbol{\theta} = \{(R_k, w_k), k = 1..L\}$
 1: function FitTreeDepthS(node, $\mathcal{D}$, depth)
 2:     // c(.) is an impurity function.
 3:     //$\mathcal{D}^L(j,t) = \{(\boldsymbol{x}_i, y_i) \in \mathcal{D} : x_{i,j} \leq t\}$
 4:     // $\mathcal{D}^R(j,t) = \{(\boldsymbol{x}_i, y_i) \in \mathcal{D} : x_{i,j} > t\}$
 5:     $(j^*, t^*_{j^*}) = \arg\min_{j=1..M,\, t \in \mathcal{T}_j} \frac{|\mathcal{D}^L(j,t)|}{|\mathcal{D}|} c\left(\mathcal{D}^L(j,t)\right) + \frac{|\mathcal{D}^L(j,t)|}{|\mathcal{D}|} c\left(\mathcal{D}^R(j,t)\right)$
 6:     if Not stoppingCriteria $\left(j^*, t^*, \mathcal{D}^R(j^*, t^*), \mathcal{D}^R(j^*, t^*)\right)$ then
 7:         return node
 8:     else
 9:         node.left = FitTreeDepthS $\left(\text{node}, \mathcal{D}^L(j^*, t^*), \text{depth} + 1\right)$
10:         node.right = FitTreeDepthS $\left(\text{node}, \mathcal{D}^R(j^*, t^*), \text{depth} + 1\right)$
11:         node.split = $\{X_{j^*} \leq t^*\}$
12:         return node
13:     end if
14: end function
```

The Algorithm 2.1 shows a template of top-down greedy learning algorithms of a decision tree. The core idea is to *locally* (inside internal node) optimise the selection of the best feature index j^* and the best threshold split $t^*_{j^*}$ that minimises a surrogate loss of the misclassification error, i.e.,

$$\min_{j=1..M,\, t \in \mathcal{T}_j} \frac{|\mathcal{D}^L(j,t)|}{|\mathcal{D}|} c\left(\mathcal{D}^L(j,t)\right) + \frac{|\mathcal{D}^R(j,t)|}{|\mathcal{D}|} c\left(\mathcal{D}^R(j,t)\right), \tag{2.5}$$

where $\mathcal{D}^L(j,t)$ (resp. $\mathcal{D}^R(j,t)$) is the subset of instances from the parent node that satisfies the condition $X_j \leq t$ (resp. $X_j > t$), $\mathcal{T}_j$ is the set of possible threshold split of the j-th feature and $c(.)$ is an impurity function. The Gini index and the Shannon entropy are well-known examples.

Let $\hat{\pi}_c = \frac{1}{|\mathcal{D}|} \sum_{x_i \in \mathcal{D}} \mathbb{1}_{\{y=c\}}(y_i)$, where C is the number of classes, then the Gini index is defined as

$$\mathbb{G}(\hat{\boldsymbol{\pi}}) = \sum_{c=1}^{C} \hat{\pi}_c (1 - \hat{\pi}_c) \tag{2.6}$$

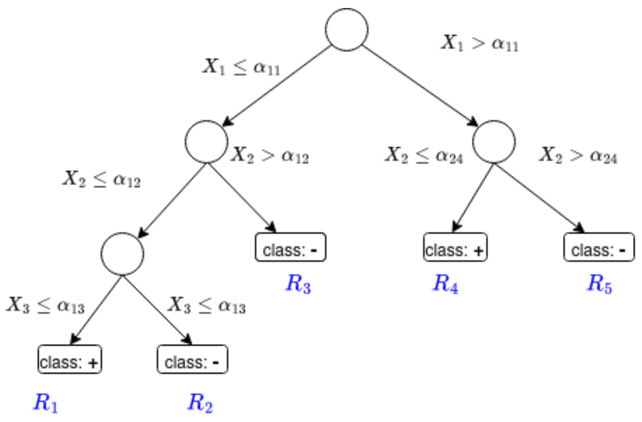


Figure 2.1: A classification tree.

and the Shannon entropy is defined as

$$\mathbb{H}(\hat{\pi}) = -\sum_{c=1}^{C} \hat{\pi}_c \log \hat{\pi}_c. \tag{2.7}$$

It is important to note that Algorithm 2.1 heuristically optimises the misclassification loss in Equation 2.4 through subsequent minimisation of impurity functions computed locally on each internal node. As analysed by Dietterich (2000) through the analogy of *boosting*, under reasonable assumptions, it has been shown that greedy approaches can provide a certain level of accuracy over training data under sufficient number of leaves. However, they cannot provide a "certificate" of optimality for a given number of leaves or depth. From the rest of the thesis, to be coherent with the literature, any method that provably learn an optimal decision tree under misclassification loss over training data will be called *optimal tree learner*.

2.2.2 Multilayer Perceptrons

Also called deep feedforward networks or feedforward neural networks, Multilayer perceptrons (MLPs) represent the *cornerstone* of deep learning.

Let $\boldsymbol{x} \in \mathbb{R}^m$ denotes an input, m being the number of features. A one-hidden-layer MLP with d_1 hidden units and d_2 outputs is described with the following equations

$$\boldsymbol{h}^{(1)} = \Phi_1 \left(\boldsymbol{W}^{(1)\,T} \boldsymbol{x} + \boldsymbol{b}^{(1)} \right) \tag{2.8}$$

$$\boldsymbol{o} = \Phi_2\left(\boldsymbol{W}^{(2)\,T}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right), \tag{2.9}$$

where $\boldsymbol{h}($1) is the first hidden representation of the input $\boldsymbol{x}$, $\boldsymbol{W}^{(1)} \in \mathbb{R}^{m\times d_1}$ (resp. $\boldsymbol{W}^{(2)} \in \mathbb{R}^{d_1\times d_2}$) is the weight of the first hidden layer (resp. of the output layer), $\boldsymbol{b}^{(1)} \in \mathbb{R}^{d_1}$ (resp. $\boldsymbol{b}^{(2)} \in \mathbb{R}^{d_2}$) is the bias of the first hidden layer (resp. output layer), Φ_1 and Φ_2 are the activation functions of the corresponding layers. Therefore, if one wants to use a one-hidden-layer MLP for the classification task, the function can be written as

$$\begin{aligned} f_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \mathrm{softmax}(\boldsymbol{o}) && (2.10)\\ &= \mathrm{softmax}\left(\Phi_2\left(\boldsymbol{W}^{(2)\,T}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right)\right) && (2.11)\\ &= \mathrm{softmax}\left(\Phi_2\left(\boldsymbol{W}^{(2)\,T}\Phi_1\left(\boldsymbol{W}^{(1)\,T}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right)\right) && (2.12)\end{aligned}$$

where $\mathrm{softmax}(\boldsymbol{z})_j = \frac{\exp z_j}{\sum_{j'}\exp z_{j'}}$ and here, Φ_2 can be the identity function.

In contrast to directly operating with input features as with *shallow* models such as decision trees, MLPs rather allow to learn more hidden and abstract-level representations $\boldsymbol{h}$ of the inputs. Another interesting propriety (though sometimes useless in practice) is their universal approximation capability for a one-hidden-layer with enough units (Hornik, 1991). Figure 2.2 shows a one-hidden-layer MLP.

A one-hidden-layer MLP in Eq. 2.10 can be rewritten in the form

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = f^{(2)}_{\boldsymbol{\theta}_2}\left(f^{(1)}_{\boldsymbol{\theta}_1}(\boldsymbol{x})\right), \tag{2.13}$$

where $f^{(2)}_{\boldsymbol{\theta}_2}(\boldsymbol{z}) = \mathrm{softmax}\left(\Phi_2\left(\boldsymbol{W}^{(2)\,T}\boldsymbol{z} + \boldsymbol{b}^{(2)}\right)\right)$, $f^{(1)}_{\boldsymbol{\theta}_1}(\boldsymbol{x}) = \Phi_1\left(\boldsymbol{W}^{(1)\,T}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right)$, $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$, $\boldsymbol{\theta}_1 = \{\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}\}$ and $\boldsymbol{\theta}_2 = \{\boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}\}$.

It is therefore possible to stack several hidden layers with a L-hidden-layer MLP through

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = f^{(L+1)}_{\boldsymbol{\theta}_{L+1}}\left(\ldots f^{(2)}_{\boldsymbol{\theta}_2}\left(f^{(1)}_{\boldsymbol{\theta}_1}(\boldsymbol{x})\right)\right), \tag{2.14}$$

Given the above MLP model, the classical way to fit this model is by considering the softmax as a categorical distribution and then optimise the negative log likelihood loss

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{N}\log p(y_i|\boldsymbol{x}_i;\boldsymbol{\theta}) = -\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbb{1}_{\{y=c\}}(y_i)\log f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)_c. \tag{2.15}$$

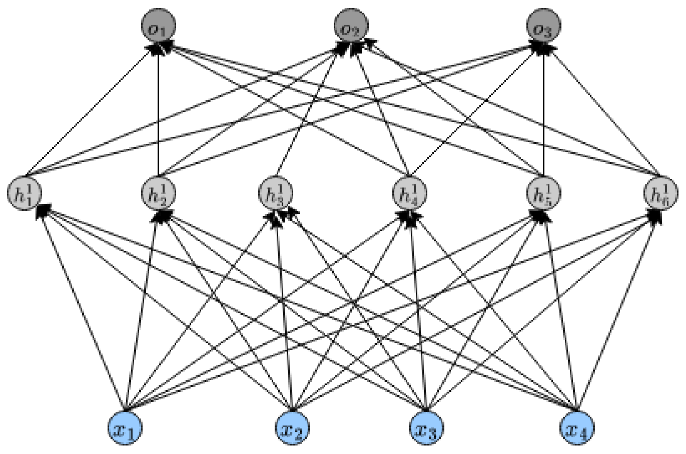


Figure 2.2: Multilayer Perceptron with one hidden layer, 4 input features and 3 outputs. Input features $\boldsymbol{x}$ are represented at the bottom of the architecture. The first hidden representations $\boldsymbol{h}^1$ are at the middle. The outputs $\boldsymbol{o}$ are represented at the top.

The rightmost term in Eq. 2.15 is called the cross-entropy loss. Since all elements in $\boldsymbol{\theta}$ have continuous values, minimising $\mathcal{L}(\boldsymbol{\theta})$ is typically done with a one-order non-linear optimisation i.e., gradient-based optimisation through an iterative process

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta_t d_t \tag{2.16}$$

where $d_t = -\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)$ is the descent direction and η_t is the step size or the learning rate at the iteration t. In practice, the gradient is computed using a chain rule over the reverse-order differentiation. This algorithm is known as *backpropagation*. Additionally, it is usually common when doing updating in Eq. 2.16 to only consider a mini-batch B of training inputs $\boldsymbol{x}_i$. This optimisation technique is called stochastic gradient descent (SGD). Finally, the stopping criteria of this optimisation process is the necessary condition of optimality in one-order non-linear optimisation, i.e., $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t) = 0$.

2.3 Discrete Latent Variable Models and EM Algorithm

This section describes (in part based on Murphy (2012)) in the context of (un) supervised [5] learning a popular type of model classes that we will use in Chapter 6.

[5] In the Chapter 1, we said that we focus on supervised learning. Discrete latent variable can be used in both supervised and unsupervised learning. For the simplicity of presentation, we describe them here in the context of unsupervised learning. However, we will use them in

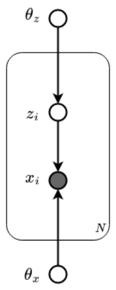


Figure 2.3: A latent variable model represented as a directed graphical model inspired from Murphy (2012). White nodes represent unobserved random variables whereas grey nodes represent observed random variables. Arrows represent the data generative process.

This type of model is the discrete latent variable model. A *latent variable* term here means that the model assumes that there are some hidden or unknown variables $\mathbf{z}$ that for example allow to generate observed variables $\mathbf{x}$. Figure 2.3 shows a graphical representation of this type of models. Note that the $\boldsymbol{\theta}_x$ and $\boldsymbol{\theta}_z$ are represented in the figure as random variables. However, in the following, they will be seen as real-valued parameters (point estimates instead of distributions).

We zoom in a particular class of model called (finite) Gaussian mixture models (GMM), where $\mathbf{z}$ follows a categorical distribution and $p(\mathbf{x}|\mathrm{z} = k, \boldsymbol{\theta})$ is a Gaussian (normal) distribution.

2.3.1 Gaussian Mixture Model

Let's suppose that we have N observations $\boldsymbol{x}_i$ that we want to infer some K abstract or hidden categories (handled through the latent variable z) over these observations. The GMM can be described with the equation

$$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} p(z_i = k|\boldsymbol{\theta}) p(\boldsymbol{x}_i|z_i = k|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{2.17}$$

where $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{m/2}|\boldsymbol{\Sigma}|^{1/2}} \exp -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k)$ is the Gaussian probability distribution function (Gaussian PDF) with mean $\boldsymbol{\mu}_k$ and the covariance matrix $\boldsymbol{\Sigma}_k$, $\pi_k (k = 1..K)$ are the mixing components of the mixture, $\boldsymbol{\theta} = \{\pi_k, \mu_k, \boldsymbol{\Sigma}_k; k = 1..K\}$.

Chapter 6 for supervised learning.

2.3.2 Model Fitting and the EM Algorithm

The log likelihood of the GMM is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_i \log \left[\sum_{\boldsymbol{z}_i} p(\boldsymbol{x}_i, \boldsymbol{z}_i | \boldsymbol{\theta}). \right] \tag{2.18}$$

Optimising directly Eq. 2.18 in its current form is difficult because of the presence of $\boldsymbol{z}_i$ that are unobserved. A first approach is to use gradient descent by taking into consideration that $\boldsymbol{z}_i$ should follow a categorical distribution i.e., $\sum_k \pi_k = 1$ and $\boldsymbol{\Sigma}_k$ should be a symmetric positive definite matrix.

A much simpler alternative is to use the expectation maximisation (EM) algorithm. If we assume a distribution $q_i(\mathbf{z}_i)$ over the hidden variable $\boldsymbol{z}_i$, then

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \textstyle\sum_i \log \left[\sum_{\mathbf{z}_i} q(\mathbf{z}_i) p(\boldsymbol{x}_i, \boldsymbol{z}_i | \boldsymbol{\theta}) / q(\mathbf{z}_i) \right] & (2.19) \\ &\geq \textstyle\sum_i \sum_{\mathbf{z}_i} q(\mathbf{z}_i) \log \left[\frac{p(\boldsymbol{x}_i, \mathbf{z}_i | \boldsymbol{\theta})}{q(\mathbf{z}_i)} \right]^{6} & (2.20) \\ &:= Q(\boldsymbol{\theta}, q), & (2.21) \end{aligned}$$

where $Q(\boldsymbol{\theta}, q)$ is called the evidence lower bound of the log likelihood. It can be rewritten as

$$Q(\boldsymbol{\theta}, q) = \sum_i \mathbb{E}_{q_i} \left[\log p(\boldsymbol{x}_i, \mathbf{z}_i | \boldsymbol{\theta}) \right] + \mathbb{H}(q_i), \tag{2.22}$$

where $\mathbb{H}(q_i)$ is the Shannon entropy of the distribution $q_i(\mathbf{z}_i)$. Therefore, one should choose the distribution q that gives the tightest lower bound. Furthermore, one can show that the i-th term of this lower bound is

$$Q(\boldsymbol{\theta}, q_i) = -D_{\mathrm{KL}}(q_i\,(\mathbf{z}_i) || p(\mathbf{z}_i | \boldsymbol{x}_i, \boldsymbol{\theta})) + \log p(\boldsymbol{x}_i | \boldsymbol{\theta}) \tag{2.23}$$

If we consider $\boldsymbol{\theta}$ to be known (let call it $\boldsymbol{\theta}^t$), since $\log p(\boldsymbol{x}_i|\boldsymbol{\theta})$ is independent of q_i, maximising $Q(\boldsymbol{\theta}, q_i)$ is equivalent to minimising $D_{\mathrm{KL}}(q_i\,\big(\mathbf{z}_i) || p(\mathbf{z}_i | \boldsymbol{x}_i, \boldsymbol{\theta}^t)\big)$ according to Eq. 2.23. This is also equivalent to set $q_i^t(\mathbf{z}_i) = p(\mathbf{z}_i | \boldsymbol{x}_i, \boldsymbol{\theta}^t)$. This step is called the expectation step (E step).

When integrating $q_i^t(\mathbf{z}_i)$ into Eq. 2.22, this equation becomes

$$Q(\boldsymbol{\theta}, q^t) = \sum_i \mathbb{E}_{q_i^t} \left[\log p(\boldsymbol{x}_i, \mathbf{z}_i | \boldsymbol{\theta}) \right] + \mathbb{H}(q_i^t). \tag{2.24}$$

[6] Using the Jensen's inequality on the log function, which is concave.

From that, since $\mathbb{H}(q_i^t)$ is independent of $\boldsymbol{\theta}$, $\boldsymbol{\theta}^{t+1}$ can be chosen to maximise $Q(\boldsymbol{\theta}, q^t)$ through

$$\boldsymbol{\theta}^{t+1} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, q^t) = \arg\max_{\boldsymbol{\theta}} \sum_i \mathbb{E}_{q_i^t}\left[\log p(\boldsymbol{x}_i, \mathbf{z}_i|\boldsymbol{\theta})\right] := \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t). \tag{2.25}$$

This is called the maximisation step (M step).

To sum up and make it simpler, in order to maximise the log likelihood in Eq. 2.18, the EM algorithm rather maximise the evidence lower bound in Eq. 2.21 by alternating the following two steps:

1. The E step: get the posterior distribution over the latent variables $\mathbf{z}_i$ by estimating

$$r_{ik} := q_i(\mathrm{z}_i = k) = p(\mathrm{z}_i = k|\boldsymbol{x}_i, \boldsymbol{\theta}^t) \tag{2.26}$$
$$= \frac{\pi_k^t p(\boldsymbol{x}_i|\mathrm{z}_i = k, \boldsymbol{\theta}^t)}{\sum_{k'} \pi_{k'}^t p(\boldsymbol{x}_i|\mathrm{z}_i = k', \boldsymbol{\theta}^t)} \tag{2.27}$$
$$= \frac{\pi_k^t \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k^t, \boldsymbol{\Sigma}_k^t)}{\sum_{k'} \pi_{k'}^t \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_{k'}^t, \boldsymbol{\Sigma}_{k'}^t)} \tag{2.28}$$

   r_{ik} values are called *responsibilities*. It is the *responsibility* that the k-*th* category of the random variable z takes to generate $\boldsymbol{x}_i$.

2. The M step: compute the parameters $\boldsymbol{\theta}$ by maximising $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t)$. One can show that for the GMM, there is a closed-form solution given by

$$\pi_k = \frac{1}{N}\sum_i r_{ik}, \tag{2.29}$$
$$\boldsymbol{\mu}_k = \frac{\sum_i r_{ik}\boldsymbol{x}_i}{\sum_i r_{ik}}, \tag{2.30}$$
$$\text{and } \boldsymbol{\Sigma}_k = \frac{1}{\sum_i r_{ik}} \sum_i (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T. \tag{2.31}$$

Of course, the step 0 is to initialise $\boldsymbol{\theta}$ before iterating.

2.4 Constraints on Machine Learning Models

For now, we have mostly defined some machine models and the optimisation tool to fit these models. However, in several cases, the problem i.e., defining a model class and fit the model that minimises the expected loss may be ill-posed or ill-defined. Indeed, the model class may be too powerful such that when fitted, it fails to generalise on unseen data. More generally, simply minimising the empirical error may lead to undesirable models such overfitted models or models that fail to meet certain requirements such as prior knowledge, robustness, etc.. This means that the model class and the objective function (e.g., empirical risk) are not enough: we need **constraints**.

Before talking about constraints in a technical way, let us define the term **constraint** by drawing inspiration from the constraint programming community.

A constraint is any relationship between variables of interest that restrains the values that can be taken by the corresponding variables simultaneously. More precisely, if we consider $v_k (k = 1..K)$ variables that can be from any domain (Boolean, integer, real, etc.), then a constraint is a relationship $R_i(v_1, ..., v_k)$ that restraints the values taken by variables $v_1, ..., v_K$. Roughly speaking, a quantity of our interest can be random variables, model parameters $\boldsymbol{\theta}$, learning algorithm $\mathcal{A}_{\mathcal{D}}(\boldsymbol{\theta})$, probability distributions, etc.

In our case, we are interested on constraints that involve the model parameters $\boldsymbol{\theta}$. It is also seen as a *prior knowledge*, which specifies a certain belief about the parameters $\boldsymbol{\theta}$ before the data has been collected. The classical source of this prior knowledge is the analyst (e.g., machine learning practitioner). But in our case, for reasons that will be described in the next chapter, we will use prior knowledge that comes from a domain-expert. For short, we will call this prior knowledge, domain-knowledge constraints. In this thesis, we will mainly focus on rich prior knowledge that have clear semantic meaning. For example, on decision trees, one may set a preference over order of selected features on decision trees to mimic the behaviour of a medical doctor for a given classification task (see Chapter 5). We will also often include the fairness constraint (see definitions in Sections 3.3.2.5 and 4.1.2) when talking about domain-knowledge constraints for the simple reason that it can be expressed as inequalities $g(\boldsymbol{\theta}) \leq 0$, where g is a function that aims to measure the (un)fairness. However, in the literature, there is no clear consensus on how to characterise fairness in the classical supervised ML setup, partially described in Section 2.1.

2.5 Brief Overview of Optimisation for Model Fitting

This sections presents an brief overview for optimisation techniques [7] that are usually used in machine learning to fit models. The description here is inspired from the books of Bertsimas and Tsitsiklis (1997), Luenberger *et al.* (1984) and Boyd *et al.* (2004). To stay coherent with the mathematical optimisation field, we will consider variables $\boldsymbol{x}$ (and sometimes $\boldsymbol{y}$). However, in machine learning, they correspond to model parameters $\boldsymbol{\theta}$ that we seek to optimise.

A mathematical optimisation problem can be defined as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f_0(\boldsymbol{x}) \\ \text{subject to (s.t.)} \quad & f_i(\boldsymbol{x}) \leq 0 \text{ and} \\ & \boldsymbol{x} \in S, \end{aligned} \tag{2.32}$$

where $\boldsymbol{x}$ is the optimisation variable of the problem, S (e.g., $\mathbb{R}^n$) is the domain of variables, $f_0 : S \longrightarrow \mathbb{R}$ is the objective function and the function $f_i : S \longrightarrow \mathbb{R}$, $i = 1..m$, are (inequality) constraint functions [8]. If all functions $f_i, i = 0..m$, are linear, then the problem in Eq. 2.32 is called a *linear optimisation* problem or simply a linear programming (LP) problem. Otherwise, the problem is said to be a nonlinear optimisation problem.

The following sections briefly give an overview some methods that deal with linear and nonlinear programming problems and that we will use in this thesis.

2.5.1 Linear Optimisation

There are several instances of linear optimisation problems. We describe here the classical linear programming problem on continuous domains and the (linear) mixed integer programming (MIP) problem.

2.5.1.1 Linear Optimisation on Continuous Domains

The classical linear optimisation problem on continuous domains can be represented in the *standard* matrix form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \text{ and} \end{aligned} \tag{2.33}$$

[7]for single objective functions

[8]Equality constraints can be also transformed into inequality constraint functions.

$$\boldsymbol{x} \geq \boldsymbol{0},$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is the matrix of constraints, $\boldsymbol{b} \in \mathbb{R}^{m \times 1}$ is the right-hand side vector, $\boldsymbol{c} \in \mathbb{R}^{n \times 1}$ is the vector coefficients of the objective function and $\boldsymbol{x} \in \mathbb{R}^{n \times 1}$ is the vector of decision variables we seek to optimise.

Note that here, the constraints are in the equality form. Inequality constraints can be transformed into this form using *slack* (additional) variables.

Basic Feasible Solution and Fundamental Theorem of LP. Let us consider the system of equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{2.34}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{A}$ is a $m \times n$ matrix. Supposing that from the n columns of $\boldsymbol{A}$, we can select a set of m linearly independent columns. For example, we suppose that the rank of A is m and we select the first m columns, resulting in a submatrix $\boldsymbol{B}$. This matrix $\boldsymbol{B}$ is nonsingular. As a result, we can obtain

$$\boldsymbol{B}\boldsymbol{x}_B = \boldsymbol{b} \quad \text{or} \quad \boldsymbol{x}_B = \boldsymbol{B}^{-1}\boldsymbol{b}. \tag{2.35}$$

for the m-vector $\boldsymbol{x}_B$ whose components are associated with the columns of submatrix $\boldsymbol{B}$. By setting $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{0})$, we set the first m components of $\boldsymbol{x}$ as the values of $\boldsymbol{x}_B$ using the same indexes of x_B and we set 0 for the other components. This leads to the following definition.

Definition 2.5.1. *(From Luenberger* et al. *(1984)) Given the set of m simultaneous linear equations in n unknowns (from Eq. 2.34), let $\boldsymbol{B}$ be any nonsingular $m \times m$ submatrix made up of columns of $\boldsymbol{A}$. Then, if all $n - m$ components of $\boldsymbol{x}$ not associated with columns of $\boldsymbol{B}$ are set equal to zero, the solution to the resulting set of equations is said to be a basic solution to Eq. 2.34) with respect to basis $\boldsymbol{B}$. The components of $\boldsymbol{x}$ associated with the columns of $\boldsymbol{B}$ denoted by the subvector $\boldsymbol{x}_B$ according to the same column index order in $\boldsymbol{B}$ are called basic variables.*

In general, Eq. 2.33 may have no basic solutions but we will avoid edge cases by making elementary assumptions regarding the matrix $\boldsymbol{A}$. We will therefore assume that $n > m$, and the rows of $\boldsymbol{A}$ are linearly independent, that is the $m \times n$ matrix $\boldsymbol{A}$ is full rank, with rank m.

Theorem 2.5.1. *(Fundamental Theorem of LP) Given a linear program in standard form (from Eq. 2.33) where $\boldsymbol{A}$ is an $m \times n$ matrix of rank m,*

- *if there exists a feasible solution, there exits a basic feasible solution;*

- *if there exits an optimal feasible solution, there exits an optimal basic feasible solution.*

The proof of this theorem can be found in Luenberger *et al.* (1984). The theorem says that it is necessary to only consider basic feasible solutions when seeking an optimal solution to a linear program. Naively speaking, one can iterate over the finite number of $\binom{n}{m}$ cases of basic feasible solutions (corresponding to the number of ways of selecting m of n columns). Of course, before trying to enumerate, one needs to check whether or not a feasible solution exists. The Farkas's Lemma (see in Luenberger *et al.* (1984)) provides conditions for that.

The Simplex Method. In contrast to enumerate all feasible basic solutions, a much simpler alternative is the well-known simplex method. It is a very old method (developped in 1940's) and has been a method of choice for several applications. However, since the past decade, advanced methods (e.g., interior point methods that we will not cover here) have challenged the simplex method.

Let us reconsider the problem in Eq. 2.33 as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad z =& \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \text{ and} \\ & \boldsymbol{x} \geq \boldsymbol{0}. \end{aligned} \tag{2.36}$$

Previously, by the fundamental theorem of LP, we said that the optimal feasible solution is an optimal basic feasible solution. This means that we can consider only basic feasible solutions. Let $\boldsymbol{x}$ be a basic feasible solution, i.e., $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$, where $\boldsymbol{x}_B$ is the vector of basic variables and $\boldsymbol{x}_N$ is the vector of nonbasic variables (each value in $\boldsymbol{x}_N$ is currently zero). Then, the objective function can be written as

$$z = \boldsymbol{c}_B^T\boldsymbol{x}_B + \boldsymbol{c}_N^T\boldsymbol{x}_N,$$

where $\boldsymbol{c}_B$ (resp. $\boldsymbol{c}_N$) are the coefficients for the basic (resp. nonbasic) variables in the objective function. Similarly, we can write the constraints as

$$\boldsymbol{B}\boldsymbol{x}_B + \boldsymbol{N}\boldsymbol{x}_N = \boldsymbol{b}, \quad \text{or} \quad \boldsymbol{x}_B = \boldsymbol{B}^{-1}\boldsymbol{b} - \boldsymbol{B}^{-1}\boldsymbol{N}\boldsymbol{x}_N \tag{2.37}$$

Substituting the second formula of Eq. 2.37 in the formula for z gives

$$z = \boldsymbol{c}_B^T\boldsymbol{B}^{-1}\boldsymbol{b} + (\boldsymbol{c}_N^T - \boldsymbol{c}_B^T\boldsymbol{B}^{-1}\boldsymbol{N})\boldsymbol{x}_N,$$

and by defining $\boldsymbol{y} = (\boldsymbol{c}_B^T \boldsymbol{B}^{-1})^T = \boldsymbol{B}^{-T}\boldsymbol{c}_B$, then z can now be rewritten as

$$z = \boldsymbol{y}^T\boldsymbol{b} + (\boldsymbol{c}_N - \boldsymbol{y}^T\boldsymbol{N})\boldsymbol{x}_N.$$

This last formula is computationally efficient and the vector $\boldsymbol{y}$ is called the vector of *simplex multipliers*. The current values of basic variables and the objective are obtained by setting $\boldsymbol{x}_N = \boldsymbol{0}$ and they are given by[9]

$$\boldsymbol{x}_B = \hat{\boldsymbol{b}} = \boldsymbol{B}^{-1}\boldsymbol{b} \quad \text{and} \quad \hat{z} = \boldsymbol{c}_B^T\boldsymbol{B}^{-1}\boldsymbol{b}.$$

Reduced Cost. Now let $\hat{c_j}$ be an entry in the vector $\hat{\boldsymbol{c}_N} := (\boldsymbol{c}_N^T - \boldsymbol{c}_B^T\boldsymbol{B}^{-1}\boldsymbol{N})$ corresponding to x_j. The coefficient $\hat{c_j}$ is called the *reduced cost* of x_j. Then, we have

$$z = \hat{z} + \hat{\boldsymbol{c}_N}^T\boldsymbol{x}_N.$$

From the equation, if the nonbasic variable x_j is assigned some nonzero value ϵ, then the objective function will change by $\hat{c_j}\epsilon$.

Optimality Test. To test the optimality, one needs to analyse what would happen to the objective function if each of the nonbasic variables were increased from zero. There are three cases

$$\begin{cases} \hat{c_j} > 0, \text{then the objective function will increase;} \\ \hat{c_j} = 0, \text{then the objective function will not change;} \\ \hat{c_j} < 0, \text{then the objective function will decrease.} \end{cases}$$

Iteration. one can see that if $\hat{x_j} < 0$ for some j, then, the objective function can be improved by increasing the value of x_j. In this case, the current basic solution is not optimal. Once a variable x_t has been selected to *enter* in the basis, one must determine which variable will *leave* the basis. It can be shown that this variable can be increased as long as all the remaining basic variables, indexed by i, remain nonnegative, i.e., this variable reaches the value

$$\hat{x_t} = \min_{1 \le i \le m} \left\{ \frac{\hat{b_i}}{\hat{a}_{i,t}} : \hat{a}_{i,t} > 0 \right\}$$

[9]The hat in b means that *a fixed iteration.*

Symplex Algorithm. Now, we present the simplex algorithm in the following steps.

1. The optimality test: compute the vector $\boldsymbol{y}^T = \boldsymbol{c}_B^T \boldsymbol{B}^{-1}$ and the coefficients $\hat{\boldsymbol{c}}_N = \boldsymbol{c}_N - \boldsymbol{y}^T \boldsymbol{N}$. If $\hat{\boldsymbol{c}}_N \geq \boldsymbol{0}$, then the current basic is optimal. Otherwise, select a variable x_t that satisfies $\hat{c}_j < 0$ as the entering variable (one heuristic is to chose the one with the smallest coefficient).
2. The pivot: compute $\hat{\boldsymbol{A}} = \boldsymbol{B}^{-1} \boldsymbol{A}_t$, the constraint coefficients corresponding to the entering variable. Find the index s that satisfies

$$\frac{\hat{b_s}}{\hat{a}_{s,t}} = \min_{1 \leq i \leq m} \left\{ \frac{\hat{b_i}}{\hat{a}_{i,t}} : \hat{a}_{i,t} > 0 \right\}.$$

   This ratio determines the leaving variable and the element called the "pivot entry" $\hat{a}_{s,t}$.
3. The update: the basis matrix $\boldsymbol{B}$ and the vector of basic variables $\boldsymbol{x}_B$. Loop again if necessary.

The previous steps describe how to run the simplex algorithm in the matrix form (with matrix inversion). In practice, there are tricks to avoid matrix inversion. There are also edge cases (unboundness, degenerate solutions, etc.) that we avoid discussing here. These edge cases are drawn when a step cannot be performed. In efficient implementations, the simplex method requires $\mathcal{O}(m^2)$ memory and $\mathcal{O}(m \times n)$ operations.

2.5.1.2 Linear Mixed Integer Optimisation

In the previous section we presented a well-known technique to solve a linear programming problem in continuous domains. When at least one variable is in discrete domains (e.g., $\mathbb{N}$, or $\{0, 1\}$), the problem becomes much more difficult and the simplex method itself cannot solve the problem. Indeed, it becomes NP-hard. This section gives a brief overview on how to solve mixed integer programming (MIP) problems. We will model two MIP problems in Chapter 5 to learn decision trees.

A general MIP problem can be formalised as

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & \boldsymbol{c}^T \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{y} \qquad (2.38) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} = \boldsymbol{b}, \end{aligned}$$

$$\begin{aligned} &\boldsymbol{y} \geq 0 \quad \text{and} \\ &\boldsymbol{x} \in \mathbb{N}^{n\times 1}, \end{aligned}$$

where $\boldsymbol{A}, \boldsymbol{B}$ are constant matrices and $\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{b}$ are constant vectors.

If there are no continuous variables $\boldsymbol{y}$, then the problem is an integer linear programming problem (ILP). If there are no integer variables $\boldsymbol{x}$, then the problem is the classical linear programming problem in continuous domains tackled in Section 2.5.1.1. Finally, if values of the integer vector $\boldsymbol{x}$ are restricted to be 0 or 1, without the vector $\boldsymbol{y}$, the problem is said to be a binary (integer) programming problem. As it can be seen, MIP problems provide a great flexibility in terms of modelling choices. But the price for efficient optimisation can be expensive.

There are three main categories of algorithms that are used to solve MIP problems.

1. Exact algorithms: they are guaranteed to find optimal solutions but may take an exponential number of iterations. They include cutting plane, branch and bound (sometimes along with cutting plane) and dynamic programming.

2. Approximation algorithms: they provide an approximate solution in polynomial time, with a bound on the degree of suboptimality.

3. Heuristic algorithms: they provide a suboptimal solution without any guarantee of the quality of the feasible solution.

In the following, we describe in a high level, the branch and bound technique, which is very used in practice. It leverages a "divide and conquer" strategy to explore feasible integer solutions. The core idea is to exploit the relaxation of the problem (defined bellow) and instead of exploring the entire feasible set, it uses bounds on the optimal cost to avoid exploring certain parts of the feasible region.

MIP Relaxation. As in common in mathematical problems, it is convenient to reduce a problem into a classical problem, for which we know efficient algorithms to solve. A relaxation problem of Eq. 2.38 is the classical linear programming problem without the integer constraints, that is

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} = \boldsymbol{b} \quad \text{and} \\ & \boldsymbol{y}, \boldsymbol{x} \geq 0. \end{aligned} \tag{2.39}$$

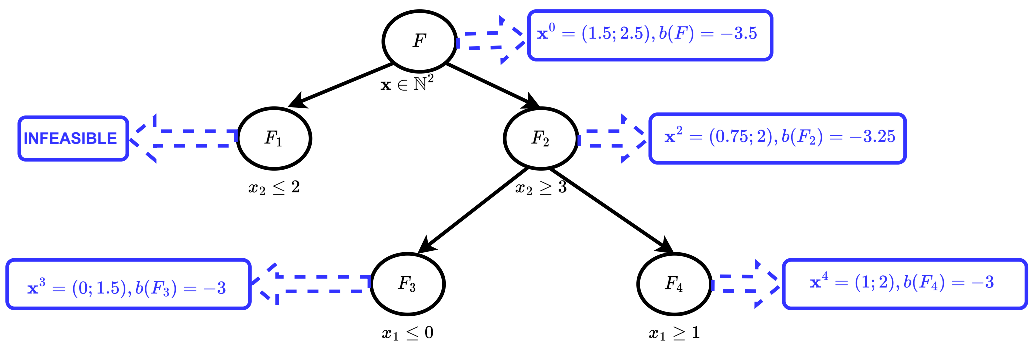

F
$\mathbf{x} \in \mathbb{N}^2$
$\mathbf{x}^0 = (1.5; 2.5), b(F) = -3.5$
INFEASIBLE
F_1
$x_2 \leq 2$
F_2
$x_2 \geq 3$
$\mathbf{x}^2 = (0.75; 2), b(F_2) = -3.25$
$\mathbf{x}^3 = (0; 1.5), b(F_3) = -3$
F_3
$x_1 \leq 0$
F_4
$x_1 \geq 1$
$\mathbf{x}^4 = (1; 2), b(F_4) = -3$

Figure 2.4: Illustration of the branch and bound method. Circles correspond to individual subproblems. Rounded rectangles show results provided by the optimal solution of the corresponding relaxation of the subproblem. This figure is inspired from the Figure 11.2 of Bertsimas and Tsitsiklis (1997).

One can see that solving this relaxation will always provide feasible solution for $\boldsymbol{y}$ but not always feasible solution for $\boldsymbol{x}$ because of the absence of integer constraints. For this reason, we will only consider from now the integer programming problem bellow

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad \text{and} \\ & \boldsymbol{x} \in \mathbb{N}^{n\times 1}. \end{aligned} \tag{2.40}$$

The branch and bound method abstracts the problem of Eq. 2.38 into the following form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{x} \in F, \end{aligned} \tag{2.41}$$

where F is the feasible set of integer solutions. Then, the method partitions the set F into k feasible sets $F_i, i = 1..k$ that correspond to k different subproblems, i.e.,

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{x} \in F_i, i = 1..k. \end{aligned} \tag{2.42}$$

The goal is to separately solve each of these subproblems and chose the best one. Each subproblem can be solved by recursively partitioning into subproblems, that is where the term "branch" comes from. Therefore, all the subproblems can be materialised with a tree as shown in Figure 2.4.

The method assumes that there is an efficient algorithm to compute a *lower bound* $b(F_i)$ for the optimal solution value [10] of the corresponding subproblem, i.e.,

$$b(F_i) \leq \min_{\boldsymbol{x} \in F_i} \boldsymbol{c}^T \boldsymbol{x}.$$

While the optimal value of the subproblem may be difficult to obtain, a lower bound is much easier. A popular method to get such lower bound is to use the optimal value of the linear programming relaxation, through e.g., the simplex algorithm!

Technically, we keep an active upper bound on the optimal value of the initial problem (non relaxed). When iterating to F_i, if the lower bound of the corresponding problem satisfies $b(F_i) \geq U$, then this subproblem can be pruned since its optimal value will be larger than the best feasible solution encountered so far. The following steps summarise the algorithm:

1. select an active subproblem F_i;
2. if the subproblem F_i is infeasible, then, delete it; otherwise compute $b(F_i)$;
3. if $b(F_i) \geq U$, then delete the subproblem;
4. if $b(F_i) < U$, either obtain an optimal solution of the problem or break the corresponding problem into further subproblems, which are added to the list of active problems.

Figure 2.4 shows a sketch of the algorithm. First, the initial problem is solved by relaxation with a solution $\boldsymbol{x}^0 = (1.5, 2.5)$, a lower bound $b(F) = -3.5$; the upper bound is $U = \infty$. Then, the problem is divided into two subproblems F_1 and F_2. The problem F_1 is infeasible, so we can delete it. Similarly as F, F_2 is divided into two subproblems F_3 and F_4. The problem F_4 is an integer feasible solution with $\boldsymbol{x}^4 = (1, 2), b(F_4) = -3$. Therefore, U can be set to -3. Since the lower bound of F_3 is $b(F_3) \geq U$, then F_3 can be deleted. As a result, $\boldsymbol{x}^4 = (1, 2)$ is the integer optimal solution.

There are several degrees of freedom in the previous algorithm. Examples of these degrees of freedom include the ways of choosing active problems, the way of obtaining lower bounds, the ways of breaking a problem into subproblems. That is why competitive MIP solvers are commercial solvers (e.g., Gurobi (Optimization,

[10]for the objective function

2021), Cplex (Cplex, 2009)) designed and maintained by several teams of researchers. However, a strong modelling (choice of variables, constraints, etc.) of a MIP problem has a great impact on the performance. A strong modelling usually requires experience, ingenuity, and art as stated by Bertsimas and Tsitsiklis (1997).

2.5.2 Nonlinear Optimisation

The previous section described classical techniques used to solve linear programming problems. This section describes popular techniques used to solve nonlinear constrained optimisation problems. We focus on convex problems on continuous domains for two reasons. First, mixed integer nonlinear problems are not used in this thesis. Second, the theory of convex problems is strong enough such that is even used for non convex problems. Before diving into the techniques, we first define the convexity of sets and functions. After, we provide some well-known techniques used to solve unconstrained problems and finally we describe techniques for constrained ones.

2.5.2.1 Convexity of Sets and Functions

Convexity of Sets. A set $\mathcal{C} \subseteq \mathbb{R}^n$ is convex if every line between two points in $\mathcal{C}$ lies in $\mathcal{C}$, i.e.,

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}, \forall \theta \in [0,1], \text{ then } \theta \boldsymbol{x} + (1-\theta)\boldsymbol{y} \in \mathcal{C}.$$

Well-known examples of convex sets are the empty set $\emptyset$, lines, line segments, euclidean balls, polyhedrons, etc.

Convexity of Functions. To define convexity of functions, one may distinguish between zero-order, first-order and second-order definitions. The order here comes from the regularity class in the theory of functional spaces.

For the zero-order condition, a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is convex if $\mathbf{dom} f$ is a convex set and

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{dom} f, \forall \theta \in [0,1], \text{ then we have } f\left(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}\right) \leq \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y}). \tag{2.43}$$

Geometrically, this inequality means that the line segment between $(\boldsymbol{x}, f(\boldsymbol{x}))$ and $(\boldsymbol{y}, f(\boldsymbol{y}))$ lies above the graph of f. The function f is concave when $-f$ is convex.

For the first-order condition, when the function f is differentiable, i.e., its gradient ∇f exists on each point of its open domain $\mathbf{dom} f$ then f is convex if

dom$f$ is a convex and

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{dom} f, \text{ we have } f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x}) \leq f(\boldsymbol{y}). \tag{2.44}$$

This equation has two meanings. First, it means that the *affine* (or first-order) Taylor approximation $f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x})$ of the function near $\boldsymbol{x}$ is a *global underestimator* of the function. Second, if the first-order Taylor approximation of a function is a global underestimator of the function, then the function is convex.

For the second-order definition, when its Hessian or second derivative $\nabla^2 f$ exists at each point in **dom**f, which is open, then f is convex if and only if **dom**f is convex and its Hessian is positive semidefinite, i.e.,

$$\forall \boldsymbol{x} \in \mathbf{dom} f, \text{ we have } \nabla^2 f(\boldsymbol{x}) \succeq 0, \text{ } i.e., , \forall \boldsymbol{y} \in \mathbf{dom} f, \boldsymbol{y}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{y} \leq 0.$$

2.5.2.2 Unconstrained Convex Optimisation

Let us consider the following optimisation problem

$$\begin{aligned} \min \quad & f_0(\boldsymbol{x}) \\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \leq 0, i = 1..m \text{ and} \\ & g_i(\boldsymbol{x}) = 0, i = 1..p, \end{aligned} \tag{2.45}$$

where $f_0 : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the objective function, $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}, i = 1..m$ are the inequality constraint functions and $g_i : \mathbb{R}^n \longrightarrow \mathbb{R}, i = 1..p$ are the equality constraint functions. Note that here we distinguish between inequality and equality constraints.

If all functions $f_i, i = 0..m$ and $g_i, i = 1..p$ are convex in their domains, then the optimisation problem of Eq. 2.45 is said to be a constrained convex optimisation problem. In this section we zoom on the unconstrained one which is simply

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}). \tag{2.46}$$

Necessary and Sufficient Optimality Condition. When the function f is differentiable and convex, it can be shown that a point $\boldsymbol{x}^*$ is optimal if and only if

$$\nabla f(\boldsymbol{x}^*) = 0. \tag{2.47}$$

Not only is this point a *local optimal*, i.e., $\exists \epsilon > 0, \forall \boldsymbol{x} \in \mathrm{B}(\boldsymbol{x}^*, \epsilon)$ [11], $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$, but this point is also a *global optimal*, i.e., $\forall \boldsymbol{x} \in \mathbf{dom} f, f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$.

[11] An Euclidean ball of centre $\boldsymbol{x}^*$ and of radius ϵ

Descent Methods. When it is not possible to analytically solve the Eq. 2.47, one of the most popular approaches to reach to the above necessary and sufficient condition are *descent methods.* Descent methods are iterative algorithms where the goal is to obtain a sequence of points $\boldsymbol{x}^{(k)}, k = 1...K$, with

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + t^{(k)}\Delta\boldsymbol{x}^{(k)},$$

where $t^{(k)} > 0$ (except when $\boldsymbol{x}^{(k)}$ is optimal) is the step size or the learning rate and $\Delta\boldsymbol{x}^{(k)}$ is a descent direction.

A descent direction means that the inequality $f(\boldsymbol{x}^{(k+1)}) < f(\boldsymbol{x}^{(k)})$ holds. It can be shown using Eq. 2.44 that when f is convex, this condition reduces to

$$\nabla f(\boldsymbol{x}^{(k)})^T\Delta\boldsymbol{x}^{(k)} < 0.$$

When the f is not differentiable, there exist zero-order methods that may assume its continuity, but we do not cover them here.

First-order Methods: Gradient Descent. As its name indicates, the gradient descent method considers the descent direction as $\Delta\boldsymbol{x}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)})$. This direction is known as the *steepest descent.* There exist several ways to find the learning rate or the step size.

- The constant learning rate uses a constant value $t^{(k)} = t$ at each step. There are theoretical guarantees of convergence for $t^{(k)} < 2/L$, where L is the Lipschitz constant of the gradient of f.
- The adaptive learning rate uses a learning rate that varies at each iteration. One way to choose this learning rate is to use the line search, which solves another optimisation problem: $t^{(k)} = \arg\min_{t>0} f\left(\boldsymbol{x}^{(k)} - t\nabla f(\boldsymbol{x}^{(k)})\right)$.

Gradient methods can be very slow to converge, especially in *flat regions.* There are advanced methods that use *momentum*, which is a way to exploit, in the current step k, not only the current gradient $\nabla f(\boldsymbol{x}^{(k)})$, but also the previous one $\nabla f(\boldsymbol{x}^{(k-1)})$.

Second-order Methods: (Quasi) Newton. Newton methods use the direction $\Delta\boldsymbol{x}^{(k)} = -\left(\nabla^2 f(\boldsymbol{x}^{(k)})\right)^{-1}\nabla f(\boldsymbol{x}^{(k)})$, which can be shown to be also a valid descent direction when f is convex. Similarly as in first-order methods, the learning rate can be a small positive constant or an adaptive one, e.g., found using linear search. In practice, even though Newton methods require very few steps to converge, they are rarely used in practice because they require matrix inversion. Quasi Newton methods such as the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method are more popular because they bypass the need for matrix inversion by approximating the Hessian matrix $\nabla^2 f(\boldsymbol{x}^{(k)})$.

Stochastic Gradient Descent. Stochastic gradient descent (SGD) is one of the most popular methods used in machine learning. As its name indicates, it is the gradient descent in *stochastic optimisation*. In stochastic optimisation, the function to minimise can be written as an average $f(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z})}\left[f(\boldsymbol{x}, \boldsymbol{z})\right]$. At each iteration k, we assume that we can sample $\boldsymbol{z}^{(k)} \sim q$ and the updating rule is

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - t^{(k)} \nabla f(\boldsymbol{x}^{(k)}, \boldsymbol{z}^{(k)}). \tag{2.48}$$

With finite sum problems, i.e., for the type of functions $f(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}, \boldsymbol{z}_i)$, there are several variants. The first ones depend on the way of computing the gradient. We can use all the finite points $\boldsymbol{z}_i$ to compute the gradient, i.e., $\nabla f(\boldsymbol{x}^{(k)}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(\boldsymbol{x}^{(k)}, \boldsymbol{z}_i)$: this corresponds to the *full batch* SGD. We can also use a *minibatch*, to compute the gradient $\nabla f(\boldsymbol{x}^{(k)}) \approx \frac{1}{\#\mathcal{B}^{(k)}}\sum_{i \in \mathcal{B}^{(k)}} \nabla f(\boldsymbol{x}^{(k)}, \boldsymbol{z}_i)$, with $\mathcal{B}^{(k)}$ a collection of $\boldsymbol{z}_i$ points at the step k.

Although the gradients computed using the minibatch are unbiased estimations, their variance do not always vanish. There is an intensive research area consisting in reducing this variance. We will mention only the most used one, which is the "adaptive moment estimation" (ADAM) method (Kingma and Ba, 2015). The ADAM method combines the momentum approach and the *preconditioned* SGD approach, which adds a preconditioned matrix similar to the inverse of Hessian to reduce the variance of gradients.

2.5.2.3 Constrained Convex Optimisation

Let us reconsider the constrained optimisation problem

$$\begin{aligned} \min \quad & f_0(\boldsymbol{x}) \\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \leq 0, i = 1..m \text{ and} \\ & g_i(\boldsymbol{x}) = 0, i = 1..p. \end{aligned} \tag{2.49}$$

The classical way to integrate the constraint in the objective function is to use the Lagrangian function $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \longrightarrow \mathbb{R}$ associated with the problem of

Eq. 2.49 defined as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i h_i(\boldsymbol{x}), \tag{2.50}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are the Lagrange multipliers for the inequality and equality constraints, respectively.

Optimality and K.K.T. Conditions. One can show that the optimisation problem defined by Eq. 2.49 is equivalent to

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda} \geq 0, \boldsymbol{\nu}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \min_{\boldsymbol{x}} \max_{\lambda \geq 0, \boldsymbol{\nu}} f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i h_i(\boldsymbol{x}). \tag{2.51}$$

This form is called the *primal form.* Using this primal version of the problem, the Karush-Kuhn-Tucker (KKT) conditions provide optimality conditions that are necessary and sufficient for convex problems. These conditions must be satisfied by the optimal solution $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$. They are listed bellow.

1. Stationary condition: the optimal $\boldsymbol{x}^*$ is a stationary point. This means that

$$\nabla f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \nabla f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i \nabla h_i(\boldsymbol{x}) = 0. \tag{2.52}$$

2. Feasibility conditions: the optimal $\boldsymbol{x}^*$ satisfies the constraints, i.e.,

$$f_i(\boldsymbol{x}^*) \leq 0, i = 1..m \text{ and } g_i(\boldsymbol{x}^*) = 0, i = 1..p. \tag{2.53}$$

3. Dual feasibility conditions: the dual values or Lagrange multipliers $\boldsymbol{\lambda}^*$ for inequality constraints are positive, that is

$$\boldsymbol{\lambda}^* \geq 0. \tag{2.54}$$

4. The complementary slackness conditions: either the inequality constraint function or the corresponding Lagrange multiplier is zero, that is

$$\lambda_i^* f_i(\boldsymbol{x}^*) = 0, i = 1..m.$$

There are cases where we can manually solve constrained optimisation problems using these conditions, e.g., constrained quadratic optimisation. However, it is unlikely to get a closed form solution using these four K.K.T. conditions.

One way to circumvent the closed form solution is to use the projected gradient descent. We will use it in Section 6.2.2 on Eq. 6.11 and we briefly describe it in the following.

Projected Gradient Descent. One of the problems with the classical descent direction when we have constraints is the fact that there is no guarantee that when iterating we stay in the feasible region. Indeed, from a current feasible (not optimal) $\boldsymbol{x}^{(k)}$ solution, just applying the descent direction to go to $\boldsymbol{x}^{(k+1)}$ does not guarantee that this solution is still feasible. The projected gradient descent *rectifies* the gradient descent direction such that the new solution continues to be a feasible solution.

It abstracts the problem of Eq. 2.49 into

$$\min_{\boldsymbol{x}} f_0(\boldsymbol{x}) \quad \text{s.t.} \quad \boldsymbol{x} \in C, \tag{2.55}$$

where C is the feasible set [12]. Supposing that at step k, we have a feasible solution $\boldsymbol{x}^{(k)}$, after applying any gradient-based update, e.g., the classical gradient method $\boldsymbol{y}^{(k+1)} = \boldsymbol{x}^{(k)} - t^{(k)} \nabla f(\boldsymbol{x}^{(k)})$, then the projected gradient descent comes to the stage by solving the constrained quadratic optimisation problem

$$\boldsymbol{x}^{(k+1)} = \arg\min_{\boldsymbol{x}} ||\boldsymbol{x} - \boldsymbol{y}^{(k+1)}||_2 \quad \text{s.t.} \quad \boldsymbol{x} \in C. \tag{2.56}$$

In most cases, with this constrained quadratic optimisation problem, we can easily get the closed form update $\boldsymbol{x}^{(k+1)}$ for $\boldsymbol{x}^{(k)}$ using the four K.K.T. conditions.

In summary for this section on optimisation for model fitting, we describe well-known techniques to solve constrained optimisation problems. We focus on single objective functions for both linear and nonlinear programming. In Chapter 5, we will use linear programming to introduce a new formalism to enforce domain-knowledge constraint on decision trees. Finally, in Chapter 6, we leverage nonlinear optimisation to integrate implicitly the explainability constraint for decision rule explanations in differentiable black-box models.

[12] $\{\boldsymbol{x} \in \mathbf{dom} f \bigcap \left(\bigcap_{i=1}^{m} \mathbf{dom} f_i\right) \bigcap \left(\bigcap_{i=1}^{p} \mathbf{dom} h_i\right); f_i(\boldsymbol{x}) \leq 0, i = 1..m, h_i(\boldsymbol{x}) = 0, i = 1..p.\}$

Chapter 3

Constraint Enforcement on Decision Trees: a Survey

This chapter study how the literature on decision trees handles constraint enforcement. We first begin by clarifying the scope and the motivation in Section 3.1. Second, Section 3.2 describes constraint acquisition or elicitation. Third Section 3.3 presents our taxonomy of constraints applied to decision trees. Finally, Section 3.4 our taxonomy of methods before macroscopically discussing all the methods in Section 3.5.

Contents

This chapter is largely based on our published paper in the journal ACM Computing Survey, entitled "Constraint Enforcement on Decision Trees: a Survey" (**Nanfack** *et al.*, 2022a).

3.1 Motivation and Scope

Let us reconsider the example in Section 1.1 where in a bank, we want an ML model whose task is to predict whether a customer should be granted credit. Similarly as a bank advisor who relies on features such as age, marital status, gender, income, loan history, etc., the ML model will be trained on data with these features. To choose suitable ML model classes, one should take into account one important prescription. Indeed, in such (finance) domain, customers may have the right to know, in human-understandable terms, why the model has reached to the decision to deny or grant credit. Therefore, we consider two classical possibilities: either we use interpretable ML models or black-box models with explainability tools.

3.1.1 Interpretable Model: Decision Tree

Decision trees are one of the most well-known, non-linear and simplest machine learning algorithms. Their representability and their ability to produce rules with relevant attributes make them the most commonly used technique when seeking interpretable machine learning models (Freitas, 2014). Additionally, they have the particularity of being ML models that are *visually* easy to understand (see Figure 3.1). Therefore, they are primarily suited for sensitive domains like medical diagnosis, finance where decisions need to be explainable.

3.1.2 Black-box Models

Being much more complex, black-box models (such as random forests and MLPs) offer the advantage of being very accurate in a wide variety of cases. However, as they are not inherently interpretable, for the example mentioned above, they should be accompanied with some explainability mechanisms. One of such mechanism is called **post-hoc** explainability, where for example, the black-box model can be approximated either locally (in a region of the input space) or globally (on all the input space) by an interpretable model such as a decision tree. This interpretable (also called surrogate or proxy) model will be used to explain predictions of the black-box model.

Let us just zoom from now on the decision tree models in both cases (case where they are the chosen model or the case where they are use as *surrogate* model of the complex one). Apart from the consideration on interpretability in our example on credit prediction, there are several other requirements or constraints that decision trees may be expected to meet. These will be detailed in the following section.

3.1.3 On the Importance of Constraint Enforcement on Decision Trees

3.1.3.1 Constraints to Improve Interpretability

When presenting decision trees and their interpretability, we omit to say that not all decision trees are interpretable. Indeed, learned decision trees can be large (having a high number of nodes) and deep (having a high depth), resulting in the loss of their interpretability. This is confirmed by the study of Piltaver *et al.* (2016), which reveals that the size of the tree, the depth and the number of leaf nodes directly impact the comprehensibility of decision trees. Thus, a desired goal would be to generate small and less deep trees (to be comprehensible) while providing a good level of accuracy. In this sense, the work of Bessiere *et al.* (2009) showed that this can can be done in some cases as they constraining the size of the decision tree can significantly improve the accuracy of traditional algorithms while halving the size of the tree.

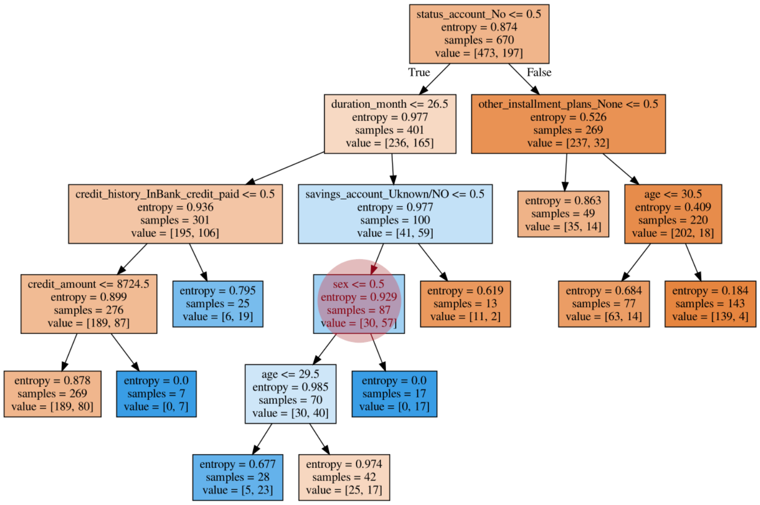


Figure 3.1: Example of a decision tree trained on the German credit dataset. Here, split rules are visible inside (root) internal nodes. This figure is taken from **Nanfack** *et al.* (2022a)

3.1.3.2 Constraints for Ethical Safeguards

In fact, when applied on our example on credit approval, an ML model may be required to ensure fair and equitable decisions (for instance, between men and women), as well as the protection of sensitive data information (called privacy). From the ethical point of view, a failure to provide this guarantee may violate the regulation in domain of finance. Figure 3.1 shows an example of decision tree learned from the well-known *German credit* dataset (Dua and Graff, 2017). While sometimes it may make unfair predictions between women and men to grant a credit (due to the selected feature *sex* highlighted in the figure), such a decision tree could be learned and used in practice if a constraint on fairness is not enforced. As ML models must meet ethical requirements like fairness or even privacy and, most importantly, one should also be able to assess that the models actually meet these requirements, if decision trees are learned with fairness and privacy constraints, they will more likely be accepted in a critical domain such as justice where the satisfaction of constraints may be more important than providing a good level of accuracy (Dziugaite *et al.*, 2020; Ribeiro *et al.*, 2016; Barredo Arrieta *et al.*, 2020). Several works (Kamiran *et al.*, 2010; Aghaei *et al.*, 2019; Liu *et al.*, 2009) attempted to impose fairness and privacy constraints on decision trees.

3.1.3.3 Constraints for Domain-related and Real-world Goals

In the field of health or banking, decision trees are widely used since people need to understand how the algorithm has reached its decisions. In the medical domain, the domain expert (*i.e.*, the doctor) validates a particular machine learning model by comparing it with his knowledge about the domain. Here (where the model is a decision tree), the doctor examines the sequence of decision rules to evaluate the comprehensibility of the decision tree. However, learned decision trees do not necessarily make sense from a medical or clinical point of view, because the algorithms only consider information that can be extracted from a medical dataset (López-Vallverdú *et al.*, 2007). Indeed, in this setting, minimising expected loss is only one among others (Cotter *et al.*, 2019). Another goal could be the fact that the learned tree should be closed as much as possible domain-related best practices. Hence, if the rules of learned trees do not match the needs of experts and their knowledge, the entire tree may be rejected. On the other hand, if decision trees are learned by considering additional domain knowledge in the form of constraints, the resulting tree could be more trustworthy and reliable. For example, works like the one of López-Vallverdú *et al.* (2007, 2012) enforce constraints on decision trees by adding priority of relevance on attributes to make trees more comprehensible and trustworthy for users and domain experts.

3.1.3.4 Constraints on Proxy or Surrogate Models

As we mention at the beginning of Section 3.1, decision trees can also be used to approximate black-box models (*e.g.*, neural networks) such that predictions can be explained. This type of models are called proxy or surrogate models (Guidotti *et al.*, 2018), and the process of approximating is often referred as *knowledge distillation*. Without any constraints such as those related to the complexity, surrogate decision trees may fail at providing clear and understandable explanations of the target black-box model. Indeed, the structure of decision trees (described in Section 3.3) might be too simple to approximate with enough *fidelity* [1] black-box models that are more complex (such as neural networks) or might be too complex, resulting in trees that have lost their interpretability. Even worse, if the black-box model embeds guarantees such as fairness, the learned tree may have little chances to meet these constraints without any constraint enforcement. For example, a fair neural network could be approximated by an unfair decision tree simply because the approximation algorithm did not incorporate the non-discrimination or fairness constraints. More dangerous, the other way around can be also be an issue: a fair

[1] agreement between outputs Y given inputs $\mathbf{X}$

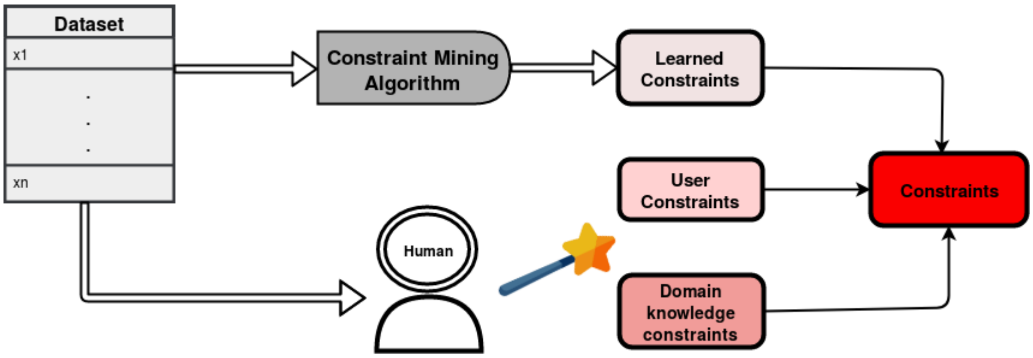


Figure 3.2: Sources of constraints. Figure taken from **Nanfack** *et al.* (2021a). Three different sources are shown: algorithms can discover and learn constraints from a given dataset; humans can define user constraints to the learning algorithm specific to their needs, for instance, to control the complexity of a tree; humans can also give constraints based on their knowledge regarding a dataset within a particular domain.

decision tree approximating an unfair neural network. This research direction is attracting a lot of interest these recent years and is being popularised under the term *fairwashing* (Aïvodji *et al.*, 2019).

At this state of the chapter, several reasons motivate the enforcement of constraints on decision trees. In the rest of this chapter, we review the literature on this topic. It is important to note that Safavian and Landgrebe (1991); Buhrman and De Wolf (2002); Lomax and Vadera (2013); Barros *et al.* (2012) review different decision tree learning algorithms. However, they lack special attention to techniques used to learn trees under constraints, which is precisely the focus of this work. Hence, it is not the scope of the rest of this chapter to present all existing tree learning methods but to review how the literature integrates constraints in the learning process to get more trustworthy decision trees.

3.2 Source of Constraints

In the previous section, we talked about the importance of enforcing constraints on decision trees. Before diving into the main review part, we begin in this section by providing an overview on some of the origins of these constraints.

Fig. 3.2 shows three major sources of constraints. First, constraints can be derived from another machine learning algorithm (e.g., rule learning algorithm) that we call hereafter constraint mining algorithm. Second, constraints may also come from users, for example they may want to limit the size of the tree. The term *user constraint* usually indicates in the literature that constraints do not require relevant domain expertise for the learning task (Garofalakis *et al.*, 2003, 2000; Fromont *et al.*, 2007). Note that machine learning practitioners may considered as users here. Third, and most importantly, constraints can be provided by domain experts with domain knowledge (also known as background knowledge (Núñez, 1991; López-Vallverdú *et al.*, 2012)) who can define the constraints related to the domain with respect to the given dataset.

In a recent work, von Rueden *et al.* (2021) focus on the prior knowledge (both user and domain-knowledge constraints) and proposes another hierarchical source for this category. These sources are: scientific and world knowledge (similar to our user-defined constraints), and expert knowledge (similar to our domain-knowledge constraints).

Note that the different sources of constraints described above are not specific to decision trees. In the following section, we describe constraints that are treated in the literature of decision trees and introduce our taxonomy of these constraints.

3.3 Taxonomy of Constraints on Decision Trees

This section presents our taxonomy of constraints inspired by Nijssen and Fromont (2010) and Struyf and Džeroski (2007) who define respectively constraints on the structure of the tree and who define instance-level constraints (especially for clustering).

In our taxonomy, we distinguish three types of constraints (see Fig. 3.3): structure-level constraints, attribute-level (or feature-level) constraints, and instance-level constraints. With this taxonomy, structure-level constraints may be set by any user to control the complexity of the decision tree. Besides, attribute-level and instance-level constraints are more susceptible to be defined by a domain expert or by a constraint mining algorithm. In the following, we present the different types of constraints and several works that have studied them in the literature.

3.3.1 Structure-level Constraints

This section gives an overview on methods that enforce structure-level constraints (a complete description of these methods can be found in our work (**Nanfack** *et al.*, 2022a)). From Figure 3.4, these constraints are defined on the size of the tree, the depth, the number of leaf nodes.

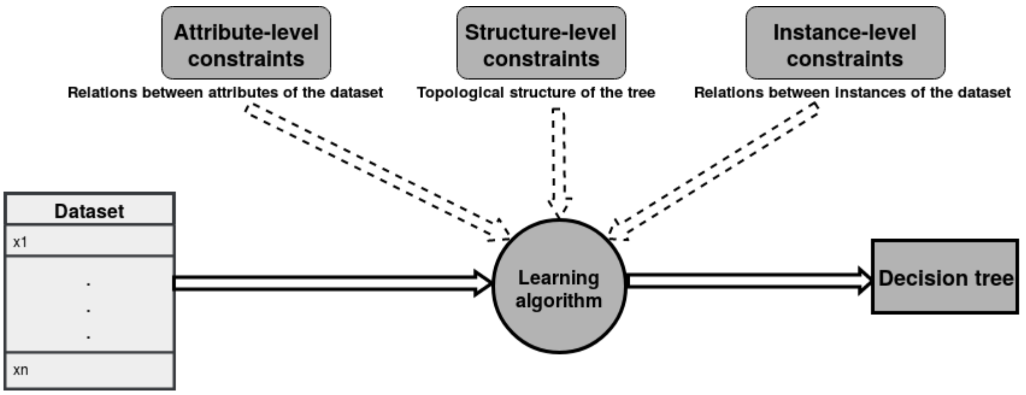

Attribute-level constraints
Relations between attributes of the dataset
Structure-level constraints
Topological structure of the tree
Instance-level constraints
Relations between instances of the dataset
Dataset
x1
xn
Learning algorithm
Decision tree

Figure 3.3: Taxonomy of constraints. Figure taken from **Nanfack** *et al.* (2022a). The dashed arrows indicate optional constraints, while solid line arrows indicate mandatory concepts. For example, a learning algorithm may integrate attribute-level constraints to learn a decision tree.

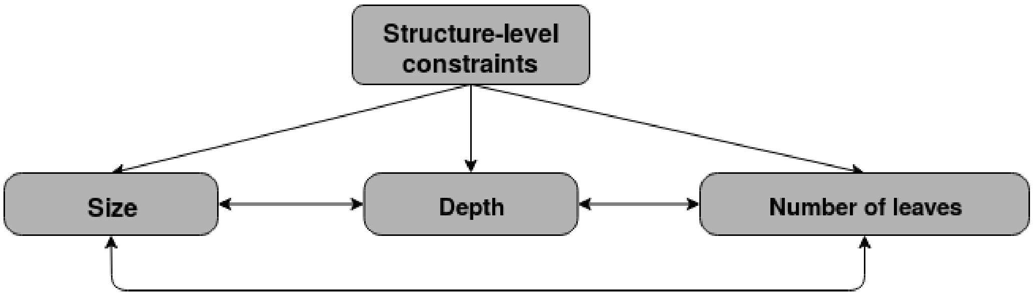


Figure 3.4: Structure-level constraints in decision trees. This figure is taken from **Nanfack** *et al.* (2022a). Structure-level constraints are refined into three sub-categories of constraints that have an impact of the structure of trees. The left and right horizontal arrows mean that the constraints have a mutual impact. For example, constraining the size of the tree limits its depth and reciprocally.

3.3.1.1 Size of the Tree

The size of a decision tree is the number of nodes of the tree and is related to the readability of the entire tree (Piltaver *et al.*, 2016). We classify works that study this constraint in three types of methods, namely top-down greedy, safe enumeration and linear (and constraint) programming methods.

Top-down greedy methods, as a reminder of Section 2.2.1 aim to optimise a (local) heuristic. Quinlan and Rivest (1989) were one of the first works which integrates the size constraint in the form of prior distribution. Drawing inspiration from the minimum description length principle, this prior distribution allows them to introduce a new loss which is optimised in a greedy top-down fashion. Latter, Garofalakis *et al.* (2000) introduce another algorithm that also integrates the size constraint via an estimation of a lower bound of the inaccuracy when deciding to split on a node using a top-down fashion. In the case of decision trees as *proxy* models, several works have also proposed to enforce the size constraint in order to ensure that decision rules explanations are not complex to grasp. Craven and Shavlik (1995) (TREPAN) and Yang *et al.* (2018) (GIRP) enforce this constraint using a post-pruning after learning proxy decision trees in a top-down greedy fashion. The loss to prune theses trees is composed of two terms: the first term is the average information gain and the second one is a penalising term on the size of the tree.

Safe enumeration methods allow to enumerate all (or a subset of) possible trees by identifying possible splitting rules with a specific attention to complexity. The size constraint permits to restrict the feasible set of potential solutions. In this direction, Bennett and Blue (1996) propose an algorithm, called global tree optimisation, which uses multivariate splits and models decision tree encoding as disjunctive inequalities with a fixed structure. The main notable works are those of Garofalakis *et al.* (2003) and Fromont *et al.* (2007) (latter extended by Nijssen and Fromont (2007)). These works leverage branch-and-bound algorithm coupled with dynamic programming to enumerate possible trees that satisfy the size constraint. While the former uses an *enumerate-and-prune* strategy whose goal is to prune an accurate and large tree such that it will satisfy the size constraint, the later uses the analogy of itemset mining with two methods: CLUS, which is a greedy method and CLUS-EX based on an exhaustive search that enumerates possibilities expecting constraints are restrictive enough to limit the search space. Latter, Nijssen and Fromont (2010) introduce a more general framework called DL8 which also uses the itemset mining approach to learn decision trees for various types of structure-level constraints (size of the tree, number of leaf nodes, *etc.*). Also based on dynamic programming, this extended version of DL8 needs enough memory to encapsulate all the lattices of variables.

Instead of safely or exhaustively enumerating trees that satisfy the size constraint, other methods prefer to use a *global* and *principled* view by formalising in terms of variables and constraints both the tree encoding and the objective function. This formalism allows (whenever applicable) to use a constraint programming (CP), integer linear programming (ILP) or satisfiability (SAT) solver to optimise the problem. Two notable works are those of Bessiere *et al.* (2009) and Narodytska *et al.* (2018). The first one proposes a method to find decision trees with minimum size using CP and ILP. Translating the problem using linear constraints and integer variables makes it possible to use ILP to explore richer and smaller trees. However, computational time remains high. To speed up the search, the second work (Narodytska *et al.*, 2018) introduced another SAT-based encoding for optimal decision trees based on tree size. The heart of this method is to consider a perfect (error-free) binary classification where the selected features on nodes and the valid tree topology are modelled with SAT formulae.

As a brief summary about the size constraint on decision trees, the majority of works enforce this constraint to better control the complexity of decision trees, making them more easily understandable and readable. The problem of learning decision trees under this constraint has been vastly explored through heuristics, enumeration and constraint programming approaches. The last two methods generally assume trees to be binary, even for non-binary categorical variables. This assumption allows to simplify the problem. However, top-down greedy approaches do not need this assumption.

3.3.1.2 Depth of the Tree

The depth constraint is important to control overfitting but also the comprehensibility of decision trees.

Top-down greedy methods can easily enforce the depth constraint using the stopping criteria in a depth-first building way. This can also be seen as a sort of *pre-pruning* strategy. Examples of works that use this approach include studies of Zilke *et al.* (2016) and Sato and Tsukimoto (2001) on proxy decision trees.

Enumeration-based methods have also been proposed to learn optimal trees (misclassification error being the objective function). For example, the T2 (Auer *et al.*, 1995) and T3 (Tjortjis and Keane, 2002) algorithms respectively find optimal trees with a maximum depth to 2 and 3 using a careful exhaustive search based on agnostic learning. Authors of T2 are one of the rare authors who proposed a constraint-based tree learning algorithm, and theoretically analyse the time complexity of the learning algorithm. T3C (Tzirakis and Tjortjis, 2017) is an improved version of T3, which changes the way T3 splits continuous attributes to four decision rules. In order to enforce the depth constraint in DL8 (presented in Section 3.3.1.1), Aglin *et al.* (2020) introduce DL8.5. This algorithm uses a branch-and-bound search with caching in order to safely enumerate trees under the depth constraint. Thanks to caching, DL8.5 is much faster than DL8.

Several works handle the depth constraint using Linear, SAT, CP approaches. Examples of mixed integer programming (MIP) include OCT (Bertsimas and Dunn,

2017) (which considers univariate and multivariate splits) and BinOCT (Verwer and Zhang, 2019) (which uses a heuristic to decrease the number splits). An example of a CP approach is the work of Verhaeghe *et al.* (2019), which uses the analogy of itemset mining of DL8 (Nijssen and Fromont, 2007). Examples of SAT approaches include the works of Narodytska *et al.* (2018), Avellaneda (2020) and Hu *et al.* (2020).

In conclusion, methods that aim to learn decision trees under depth constraints are generally based on enumerations, linear programming, and constraint programming. Their principal goal is to accelerate computations required to reach the most accurate and depth-constrained decision trees. Because these methods control the depth of the tree, they usually learn decision trees with depth at most 2, 3 or 4. However, for interpretability there is an undesirable risk of *over-simplicity* of decision rules as discussed in Section 3.5.

3.3.1.3 Number of Leaf Nodes

The number of leaf nodes is a major factor in the summary of the decision rules, in particular when trying to understand how the model predicts a particular class (Piltaver *et al.*, 2016). In the case of binary trees, the number of leaf nodes L is linked to the size of the tree $|V|$ by the formula $|V| = 2L - 1$. In other general cases, there is no such explicit formula. However, the leaves and nodes do not relate to the same aspects in terms of the explainability of decision trees. The size of the tree is related to the readability of the tree while the leaves along with decision paths is essential for the comprehensibility of the predictions of a given class. Very few studies have focused their interest on constraining the number of leaves (due to the classical preference for binary trees).

By drawing inspiration from Angelino *et al.* (2018) (optimal rule lists), the work of Hu *et al.* (2019) is only one of the rare safe enumeration approaches that explicitly constrain the number of leaves on decision trees. The framework called optimal sparse decision trees (OSDT) (latter extended to a generalised version, GODST) uses a branch-and-bound search, in the frame of structured risk minimisation (objective function with a regularisation term over the number of leaves). OSDT and GOSDT can learn very sparse decision trees that are likely to generalise well.

As linear and constraint programming formulations are used to learn decision trees under size and depth constraints, they can also be adapted for a constraint over the number of leaves. Examples of such adaptations are OCT (Bertsimas and Dunn, 2017) and the work of Menickelly *et al.* (2016).

In a probabilistic view of decision trees, the explicit modelling of the leaves is usually unavoidable. That is why almost all the proposed (Bayesian included) probabilistic methods allow to constraint via a prior distribution, the number of leaves. One of the first work in this direction is the one of Chipman *et al.* (1998). They present a Bayesian approach to learn decision trees. Their main contribution is to propose a prior over the structure of the tree (taking into account the number of leaves) and a stochastic search of the posterior to explore the search space and therefore find some *acceptable* trees. The search consists in building a Markov Chain of trees by the Metropolis-Hasting algorithm considering the transitions: *grow* (*i.e.*, split a terminal node), *prune* (*i.e.*, choose a parent of terminal node and prune it), *change* (*i.e.*, change the splitting rule of an internal node), and *swap* (*i.e.*, swap splitting rules of parent-child pair), all randomly. Extensions of this work of Chipman *et al.* (1998) have been proposed latter. Examples are those of i) Angelopoulos and Cussens (2005b), which exploits stochastic logic programming (SLP) to integrate informative priors that try to penalise unlikely transitions, ii) Angelopoulos and Cussens (2005a), which improves the convergence and predictive accuracy, and iii) Nijssen (2008), which adapts the DL8 algorithm (Nijssen and Fromont, 2007) in a Bayesian setting. Note that in all these Bayesian formulations, a single tree can be removed through the MAP.

As a summary for the number of leaves, the majority of works on the constraint over the number of leaves are Bayesian methods. They integrate this constraint via a prior distribution.

3.3.1.4 Summary and Discussion about Structure-level Constraints

To sum up, we presented works that learn decision trees under the number of leaves or the depth or the tree size constraint. The motivation varies greatly between these works. Indeed the motivation can be to search for more interpretable trees, more or less complex, or more accurate or even to speed up the learning. Also, these structural characteristics (size, number of leaves, depth) of the tree are linked to each other, because setting one value constrains the others. However, they have very different semantic meaning with respect to the comprehensibility of decision tree according to the study of Piltaver *et al.* (2016).

It is also worth noting that they are other constraints like the tree balance constraint (also depending on the branching factor of the tree), which is understudied in the literature, despite its impact on the comprehensibility of decision trees. Also, the majority of the works on structure-level constraints assume that decision trees are binary to better handle the number of leaf nodes and the sizes. This assumption is severely lacking in flexibility. In some cases, for the sake of

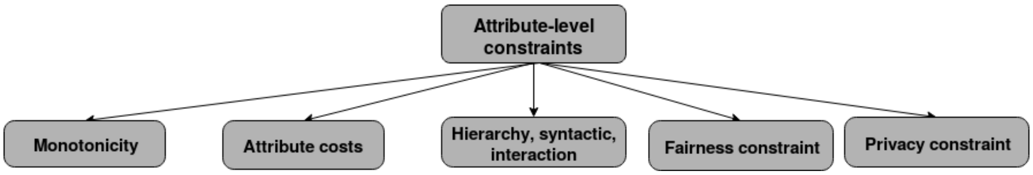


Figure 3.5: Attribute-level (or feature-level) constraints in decision trees. The top-down arrow shows the different sub-categories of constraints of attribute-level constraints. Figure taken from **Nanfack** *et al.* (2021a).

interpretability, it would be useful to have nodes with more than two child nodes. For example, if a categorical variable like the number of doors of a car has 3 values (namely 2, 4 and 6); it may be important, for comprehensibility purpose, not to transform this variable into 2 binary variables so that the knowledge "number of doors" appears only once in a branch of the tree. Additionally, another generally common assumption is that trees with small depth and small trees (small size) enhance the interpretability and the comprehensibility. This question requires further study because, actually, in certain domains such as health care, a decision tree with a maximum depth of two like in papers of Tjortjis and Keane (2002) and Bertsimas and Dunn (2017) could not be comprehensible by experts (Freitas, 2014; López-Vallverdú *et al.*, 2007). Indeed, rules may be over-simplistic to fit domain-expert requirements.

3.3.2 Attribute-level Constraints

Attribute-level constraints are defined as a relationship over the features of the dataset and decision rules of the tree. These constraints are more likely to come from an expert or a constraint mining algorithm. Most of the works on attribute-level constraints study monotonicity, attribute costs, hierarchy constraints including interactions between multiple features, privacy, and fairness (see Fig. 3.5).

3.3.2.1 Monotonicity Constraints

Monotonicity constraints are related to attributes that evolve in the same direction as the output variable. More formally, it is defined by Pei *et al.* (2016) as follows: given a set of attributes $\mathcal{A} = \{\mathbf{X_1}, ..., \mathbf{X_M}\}$, one of its subset $\mathcal{B} = \{\mathbf{X_k}, ..., \mathbf{X_l}\}$ with $1 \leq k \leq l \leq m$, the decision rule $T_{\mathbf{X}}$ of a tree T is monotonically consistent in terms of $\mathcal{B} \subseteq \mathcal{A}$, for a set $\mathcal{U}$ of examples drawn from the dataset $\mathcal{D}$ if

$$\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{U}, \mathbf{x}_i \preceq_{\mathcal{B}} \mathbf{x}_j \implies T_{\mathbf{X}}(\mathbf{x}_i) \leq T_{\mathbf{X}}(\mathbf{x}_j)$$

where $\preceq_{\mathcal{B}}$ defines a partial relation order on the instances regarding the subset of attributes $\mathcal{B}$.

For instance, let's suppose a dataset for a loan credit classification problem where the decision is whether to grant credit or not. One would like the decision rules of the learned tree to be monotonically consistent in terms of the income on the entire dataset (in this context: $\mathcal{U}$ is the entire dataset, $\mathcal{A}$ is the set of attributes of the dataset and $\mathcal{B}$ is the singleton of the income attribute).

Classification with monotonicity constraints is an important and well-studied problem since monotonicity improves the comprehensibility of classifiers and therefore increases the trustworthiness of machine learning models in some applications (Freitas, 2014).

Studies have been done to enforce monotonicity constraints on decision trees. Potharst and Feelders (2002) have proposed a survey about decision tree methods and this type of constraint. The majority of works that enforce the monotonicity constraint on decision trees usually employ top-down greedy algorithms. They mainly take two directions: introduction of *monotonicity-aware* impurity function as the local heuristic and post pruning.

Examples of works that introduce a *monotonicity-aware* impurity function include the work of Ben-David (1995), which proposes the *total ambiguity score*, which has two components: the first component is a non-monotonicity score based on a matrix representing a non-monotonicity constraint over branches of the trees and the second component is the ID3 score based on information theory. Other examples of this score-based constraint enforcement are those of Marsala and Petturiti (2015) which propose three measures using the notion of *dominance rough set* to integrate monotonicity constraints. An additional example is the one of Hu *et al.* (2012), which develops the rank entropy, which is still based on dominance rough sets but performs better than the rank mutual information.

For post-pruning methods, Feelders and Pardoel (2003) develop several fixing methods that aim to make monotone sub-trees using overfitted trees. Similar work is done in Cardoso and Sousa (2010), where a relabelling technique (class assignment step) enforces the monotonicity of the tree after a tree has been learned from a greedy algorithm such as C4.5. In the same logic, Kamp *et al.* (2009) propose to relabel non-monotone leaf nodes, then prune the learned tree using properties of isotonic functions, so that the final tree will satisfy monotonicity constraints.

3.3.2.2 Feature or Attribute Costs

Another constraint that is related to the general topic of cost-sensitive decision trees, is the feature or attribute costs. One motivation to enforce this constraint, in particular in the medical domain, is the fact that in this domain, medical doctors usually provide appropriate diagnostic by taking into account economic constraints (e.g., preference over certain medical tests). The aim of cost-sensitive decision trees (Lomax and Vadera, 2013) is to learn decision trees with a good trade-off between the misclassification error/cost and the sum of the cost of attributes that can be expressed as constraints. The survey of Lomax and Vadera (2013) categorises different methods into greedy and non-greedy. Greedy methods can use the cost during the tree learning by modifying heuristics (Tan, 1993; Pazzani *et al.*, 1994; Norton, 1989; Núñez, 1991; Ling *et al.*, 2004; Davis *et al.*, 2006; Freitas *et al.*, 2007; Li *et al.*, 2015; Wang *et al.*, 2018) or by post-pruning (Pazzani *et al.*, 1994; Ferri *et al.*, 2002), which necessitate relabelling techniques. Non-greedy methods are usually based on genetic algorithms (Turney, 1995; Omielan and Vadera, 2012; Krętowski and Grześ, 2006), wrapper methods (Estruch *et al.*, 2002) and stochastic search methods (Esmeir and Markovitch, 2004, 2006).

Enforcing cost constraints on decision trees makes them more *natural* and reliable for practical applications (Qiu *et al.*, 2017), such as loan credits in finance or medical diagnoses in the health domain.

3.3.2.3 Hierarchy and Feature Interaction

The concept of hierarchy defines any type of ordering between variables selected on a decision tree. It is also mentioned in Fromont *et al.* (2007) as syntactic constraints and in Nijssen and Fromont (2007) as path constraints. Hence, in some cases, it may simply be expressed as an attribute that must be selected before another one over a branch or even over the entire decision tree. Some domain knowledge can refer to such kind of preferences where some attributes should be tested before others hence affecting the hierarchy to be learned. For instance, in the medical domain, doctors might want to perform temperature checks or blood checks before going after more advanced tests. As a result, it makes the decision tree more reliable and more comprehensible for domain experts.

The majority of works that tackle this constraint usually use top-down greedy algorithms. Three main directions exist: either modifying the impurity function to embed the hierarchy, either modifying the list (e.g., by weighting) potential features for the split or use a sparsity constraint for multivariate decision trees. In the direction of modifying the impurity function, Núñez (1991) was one of the first works in this direction (because the issue was presented just a few years after ID3 and CART were introduced to the community). The purpose is to make decision trees more compliant with background knowledge using ISA (Is A) relationships. ISA relationships represent a hierarchy relationship between two attributes: one that belongs to the concept of the other (for example, an amphibian is a vertebrate). Núñez (1991) introduces a cost-sensitive measure that incorporates ISA relationships between variables to make the tree more comprehensible from a user perspective. Another example of works, but more applied, using this direction, is the work of Kruegel and Toth (2003), which proposes a modified version of entropy that is designed to be aligned with preexisting rules in the context of intrusion detection in computer networks.

In the direction of modifying or adjusting the potential list of features that will be considered for splits, examples of works include López-Vallverdú *et al.* (2007); Torres *et al.* (2011); López-Vallverdú *et al.* (2007), where priorities and relevance of features are formalised using healthcare criteria (in the medical domain).

Finally, Norouzi *et al.* (2015) and Madzarov *et al.* (2009) are works that enforce a sparsity constraint in multivariate decision trees.

3.3.2.4 Privacy Constraints

Privacy constraints guarantee that the learning algorithm may not access, discover, or use sensitive information to learn or predict (Vaghashia and Ganatra, 2015). Hence, it is related to both computer security and machine learning.

Before the emergence of differential privacy, various algorithms have been proposed in the literature to deal with privacy constraints in machine learning. Examples of works (using a top-down greedy approach), from the security perspective, are described bellow.

Du and Zhan (2002) propose a method to compute an impurity measure from two vertical/horizontal partitions of datasets using an untrusted third party. In the logic of horizontal partitioning data with privacy constraints, Gangrade and Patel (2012) propose a secure protocol with another untrusted third-party to learn decision trees on horizontal partitions of the dataset. In the setting of federated learning, Li *et al.* (2020a) propose a framework to learn decision trees and gradient boosting trees using hashing. Interestingly, they prove that their framework satisfies the privacy model. Friedman *et al.* (2006) propose the integration of the k-anonymisation (Sweeney, 2002) (a process to obtain an anonymous dataset while maintaining the usefulness of the data) method into the learning process of the decision tree. Teng and Du (2007) come with a hybrid approach that uses both k-anonymisation and secure multiple channels. Alternatively, Zaman *et al.* (2016) propose a method that generalises values of sensitive attributes to anonymise the dataset.

In the direction of adapting the local heuristic, Vaidya *et al.* (2008) and Gangrade and Patel (2009) present a modified version of ID3 and C4.5 respectively with a secure channel to exchange sensitive features. After proposing a new perturbation method of the training data, Liu *et al.* (2009) present a modified version of C4.5 based on a new estimation of information gain with noisy perturbed data. As this version of C4.5 results in loss of performance, Li *et al.* (2020b) introduce two algorithms based on differential privacy to guarantee privacy on decision trees and gradient boosting trees, which alleviate the performance degradation.

Other works like Matwin *et al.* (2005) prefer early stopping and pruning methods to check the satisfiability of privacy constraints during the tree learning process.

As a brief summary, learning decision trees under privacy constraints becomes an issue as soon as preserving both the privacy and the comprehensibility of the tree is needed. Aside from secure multi-channel and anonymisation methods, works in the sense of applying perturbations with random noise, and most importantly differential privacy (Friedman and Schuster, 2010), seem increasingly promising. The reason is that these directions propose a rigorous mathematical way to tackle privacy constraints without focusing on security aspects.

3.3.2.5 Fairness Constraint

Another constraint that can be considered as an attribute-level constraint is the fairness constraint because it is related to one or several sensitive attribute or features. There are several fairness notions and definitions in the literature. The growing literature regarding fairness in machine learning is converging towards two types of fairness definitions: group and individual fairness (Chouldechova and Roth, 2020; Zhang *et al.*, 2016). Group fairness aims the machine learning model to treat (or predict) similarly members from different groups (according to a sensitive feature such as sex or gender). Individual fairness pays more attention to a similar treatment for two similar individuals from different groups. While not very studied, this type of fairness is somewhat also related to instance-level constraints (see Section 3.3.3).

To be more concrete, in binary classification, let us consider a decision tree T whose task is to decide to grant a credit or not, given customer data (which contains a binary-valued sensitive feature Z, e.g., sex). Through the view of conditional independence, we provide three most popular group fairness definitions according to Zhang and Long (2021); Zafar *et al.* (2017); Chouldechova and Roth (2020).

The first and most studied fairness notion is the demographic (statistical) parity and T satisfies the demographic parity or does not suffer from disparate impact if the granting credit event ($T = 1$) does not depend on the sensitive feature, i.e., $p(T = 1|Z = 1) = p(T = 1|Z = 0)$. The second one is the disparate mistreatment, which says that the misclassification event should be independent of the sensitive feature. In other words, misclassification levels (assessed in terms of overall accuracy, false positive rate, etc.) should be the same between sensitive groups, i.e., for example $p(T \neq Y|Z = 1) = p(T \neq Y|Z = 0)$ (in the case of overall accuracy, where Y is the true target). Introduced by Hardt *et al.* (2016), the last well-known group fairness notion is the *equalised odds*, which says that the granting credit event should be independent of the sensitive feature Z given Y, i.e., $p(T = 1|Z = 1, Y) = p(T = 1|Z = 0, Y)$.

Several works have been proposed to deal with unfairness in machine learning, mostly on differentiable models. On non-differentiable models such as decision trees, one of the first work is the one of Kamiran *et al.* (2010), who propose two methods to learn fair decision trees. The first one uses a modified version of information gain that incorporates the influence of a split on the discrimination. The second one is a relabelling technique that minimises the discrimination of a learned decision tree using the empirical joint distribution between the sensitive attribute and the class attribute on all the leaf nodes. Recently, in the same approach of top-down greedy methods, Raff *et al.* (2018) adapted the learning algorithm of a C4.5 decision tree to comply with fairness constraints by introducing two measures of fairness in the case of either categorical sensitive features or continuous ones and a new fair heuristic gain measure. Aghaei *et al.* (2019) also proposed a more general framework that encapsulates the learning problem of optimal fair decision trees through MIP. Their formulation deals with classification and regression problems and integrates the fairness constraint as a regularisation term of the misclassification error or mean absolute error. However, this optimal formulation needs hours to provide good results, unlike the heuristic-based method of Raff *et al.* (2018).

3.3.2.6 Summary about Attribute-level Constraints

In summary, we have sub-categorised attribute-level constraints that are the most studied in literature into monotonicity constraints, cost of attributes, hierarchies, privacy constraints or fairness constraints. Monotonicity constraints and to some extent attribute costs and hierarchies constraints may be useful to learn trees that can be more comprehensible and trustworthy for the domain expert (You *et al.*,

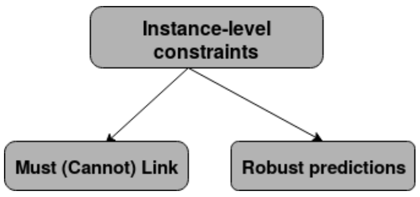


Figure 3.6: Instance-level constraints in decision trees. Figure taken from **Nanfack** *et al.* (2022a).

2017). The majority of works that deals with these constraints adapt traditional heuristics measures that incorporate constraints (Hu *et al.*, 2010, 2012; Marsala and Petturiti, 2015; Pei *et al.*, 2016). Other prefer using a post-pruning method.

The privacy constraints, which aim at preventing attacks (*e.g.*, to discover sensitive information) over instances, are generally considered through the lens of computer security. Thus, works in this field can be seen in the pure security formulations with secure channels, perturbations, anonymisation (Fletcher and Islam, 2019). However, recently, differential privacy is emerging thanks to clear mathematical formulations whose purpose is to guarantee that computations are (slightly) invariant to (noisy) perturbations over data (Friedman and Schuster, 2010). The main difficulty here is to guarantee the balance between accuracy, privacy requirements, and interpretability of the decision tree. For example, finding an optimal depth while satisfying the differential privacy constraint is still an open issue (Fletcher and Islam, 2019).

Finally, fairness constraints that act as non-discrimination constraints are widely understudied for decision trees, while being of particular interest in differentiable machine learning models. In fact, very few works exist (Kamiran *et al.*, 2010; Raff *et al.*, 2018) even though diverse mechanisms in machine learning have been proposed (Zafar *et al.*, 2017). This is partly because standard decision tree algorithms do not optimise directly a global objective and thus it is quite difficult to apply these methods to learn discrimination-aware decision trees. That is why learning fairer decision trees could be easily handled with constraint programming methods that optimise explicitly a global objective function, as proposed in Aghaei *et al.* (2019).

3.3.3 Instance-level Constraints

Instance-level constraints (see Fig. 3.6) are constraints defined through a relationship between decision trees and two or more data instances. For example one can specify that certain examples may never be misclassified. Another example of such constraints is related to clustering and thus clustering trees: instance-level constraints, such as must link or cannot link (Wagstaff *et al.*, 2001a), specify that some examples must belong to the same class or must belong to different classes. These constraints may be useful in when training data is of limited size (number of instances), and therefore can improve the performance of decision trees. This type of constraint is not also widely enforced in the literature of decision trees. In the following, we briefly describe works that try to enforce constraints of this type on decision tree.

The first constraints that are popular in clustering are must-link and cannot-link constraints. Since decision trees are mainly used for classification and regression, these constraints are not widely studied in the decision tree literature. Struyf and Džeroski (2007) propose an algorithm called ClusILC, which learns clustering trees by integrating domain knowledge in terms of instance-level constraints (must-link or cannot-link). These constraints are treated as soft constraints since some of them can be violated. They are integrated via an augmented term (which is the percentage of violated constraints) on a global heuristic that is the average variance instances over leaf nodes normalised by the total variance.

Instance-level constraints can also be specified over measures (for example information gain) directly linked to certain instances. This is the case for Sethi and Sarvarayudu (1982), who propose an algorithm that learns top-down induction trees for classification by using the average information gain on the tree, given a constraint over the probability of error. This probability of error integrates a belief over the number of instances that may be misclassified.

The last instance-level constraint, which is very popular in machine learning, is the adversarial robustness constraint. Although adversarial examples applied on decision trees are very understudied in the literature, decision trees have shown to be naturally sensitive to adversarial examples (Cheng *et al.*, 2019; Kantchelian *et al.*, 2016; Papernot *et al.*, 2016; Chen *et al.*, 2019). To overcome this problem, Chen *et al.* (2019) proposes a robust version of information gain that can serve to improve the robustness of decision trees learned in a greedy top-down fashion. Calzavara *et al.* (2020) rather optimise an evasion-aware loss function as the heuristic and Calzavara *et al.* (2019) show how to extend adversarial training for decision trees and gradient boosting trees.

3.3.4 Summary over the Taxonomy

We have described a taxonomy of constraints (tree structure, attribute, and instance levels) that are important when one may learn decision trees that comply with certain constraints. These levels seem orthogonal at first, but they are they are not because a constraint on a measure (for instance information gain must be less than a threshold) can be seen as a mixture of attribute, instance-level or structure-level constraints. As another example, imposing that the number of instances inside a leaf node is more than a fixed number (which is a very useful constraint to reduce overfitting) can be seen as instance-level and structure-level constraint. So rather than being orthogonal disjoint types of constraints, this taxonomy is flexible, since a particular level of constraints can potentially impact other levels of constraints. Hence, more constraints can be formulated as a mixture of different levels of constraints.

3.4 Taxonomy of Methods through the Lens of Optimisation

This section describes our categorisation of approaches that learn constrained decision trees. Our categorisation is partially based on the optimisation tool employed to learn decision trees under constraints. In this categorisation, we distinguish four approaches: top-down greedy, safe enumeration, linear and constraint programming, and probabilistic (sometimes referred here as discriminative and Bayseian) approaches. Table 3.1 provides an overview of these approaches for the different types of constraints.

Table 3.1: Overview of the approaches for the different types of constraints. The table can be read horizontally to look for works using a particular optimisation method or vertically to look for works that enforce a specific type of constraints.The cells of the table indicate works given the type of constraints in the header and the optimisation method in the table row. This table is taken from **Nanfack** *et al.* (2022a).

	Structure-level	**Attribute-level**	**Instance-level**
Greedy top-down	Quinlan and Rivest (1989); Craven and Shavlik (1995); Boz (2002); Garofalakis *et al.* (2000); Zilke *et al.* (2016); Wu *et al.* (2016); Yang *et al.* (2018)	Norton (1989); Núñez (1991); Tan (1993); Pazzani *et al.* (1994); Ben-David (1995); Potharst and Feelders (2002); Ferri *et al.* (2002); Daniëls and Velikova (2003); Kruegel and Toth (2003); Feelders and Pardoel (2003); Ling *et al.* (2004); Davis *et al.* (2006); Freitas *et al.* (2007); López-Vallverdú *et al.* (2007); Kamp *et al.* (2009); Cardoso and Sousa (2010); Iqbal *et al.* (2012); López-Vallverdú *et al.* (2012); Li *et al.* (2015); Wang *et al.* (2018); Gangrade and Patel (2012); Vaidya *et al.* (2008); Gangrade and Patel (2009); Matwin *et al.* (2005); Friedman *et al.* (2006); Sweeney (2002); Teng and Du (2007); Zaman *et al.* (2016); Liu *et al.* (2009); Li *et al.* (2020b,a); **Nanfack** *et al.* (2021a)	Sethi and Sarvarayudu (1982); Gehrke *et al.* (1999); Struyf and Džeroski (2007); Kamiran *et al.* (2010); Raff *et al.* (2018); Chen *et al.* (2019); Calzavara *et al.* (2020, 2019)

	Structure-level	**Attribute-level**	**Instance-level**
Safe enumeration methods	Bennett and Blue (1996); Tjortjis and Keane (2002); Garofalakis *et al.* (2003); Struyf and Džeroski (2006); Fromont *et al.* (2007); Kocev *et al.* (2007); Auer *et al.* (1995); Tzirakis and Tjortjis (2017); Lin *et al.* (2020); Hu *et al.* (2019)	Turney (1995); Estruch *et al.* (2002); Esmeir and Markovitch (2004, 2006); Madzarov *et al.* (2009); Omielan and Vadera (2012); Krętowski and Grześ (2006)	Gehrke *et al.* (1999); Nijssen and Fromont (2010, 2007)
Linear, SAT and constraint programming	Heidenberger (1996); Bessiere *et al.* (2009); Verwer and Zhang (2017); Bertsimas and Dunn (2017); Menickelly *et al.* (2016); Narodytska *et al.* (2018); Firat *et al.* (2020); Verwer and Zhang (2019); Aghaei *et al.* (2019); Verhaeghe *et al.* (2019); Aghaei *et al.* (2021); Avellaneda (2020); Hu *et al.* (2020)	Bennett and Mangasarian (1992); Aghaei *et al.* (2019)	Bertsimas and Dunn (2017); Verhaeghe *et al.* (2019)
Probabilistic learning	Buntine (1992); Chipman *et al.* (1998); Denison *et al.* (1998); Angelopoulos and Cussens (2005a,b); Wu *et al.* (2007); Schetinin *et al.* (2007); Nijssen (2008); Nuti *et al.* (2019)	Angelopoulos and Cussens (2005b,a)	Chipman *et al.* (1998); Denison *et al.* (1998); Buntine (1992); Angelopoulos and Cussens (2005b,a); Nuti *et al.* (2019)

Table 3.1: Overview of the approaches for the different types of constraints.

3.4.1 Top-down Greedy Approaches

Methods that learn decision trees in a top-down manner consist in choosing, at each node the test that optimises a specific heuristic (generally a local heuristic, which depends on a subset of data and a subset of attributes).

The majority of constraints talked in previous sections can be incorporated into a heuristic to build trees greedily. They make sure to push local decisions towards a trade-off between satisfying the stated constraints and the expected accuracy of the tree. If top-down greedy approaches have the advantage of being fast to compute, they have the big disadvantage of providing usually sub-optimal decision trees, which may be harmful when constraints need to be impose. Since they do not optimise directly a global objective, they are not naturally suited for global constraints (for example, the size of the tree or monotonicity constraints for monotone datasets). This is why post-strategies such as pruning methods are often applied to make sure that the global constraints are satisfied even when these constraints can be integrated into the heuristic (Garofalakis *et al.*, 2003; Kamiran *et al.*, 2010; Choi *et al.*, 2016). For structure-level constraints, few works handle structure-level constraints with top-down greedy approaches, as they are naturally designed to learn the most accurate trees even if the resulting trees can be large. Works that leverage a top-down approach are generally an extension of pruning methods (e.g., Quinlan and Rivest (1989); Craven and Shavlik (1995); Boz (2002); Garofalakis *et al.* (2000); Zilke *et al.* (2016); Wu *et al.* (2016)). They try first to guarantee a level of accuracy by learning possible large trees and, second, to enforce structure-level constraints using a (post) pruning strategy. This explains why they can be seen as pruning methods.

To enforce attribute-level constraints, top-down greedy algorithms are very popular for two reasons. First, the main algorithm is generic enough to be easily customisable. Second, in real-world applications, domain experts provide domain knowledge than can be transposed to attribute-level constraints. Then, practitioners can easily develop models that comply with domain knowledge.

3.4.2 Safe Enumeration Approaches

Apart from greedy algorithms, diverse works try to enumerate the possibilities of choosing splits through careful directives with respect to the constraints, while simultaneously proposing mechanisms to break the complexity (e.g., dynamic programming or branch and bound). Beyond the brute force method, methods exist to choose the best split criterion randomly according to a minimum/maximum number of possibilities. Even though these methods have the advantage of leaving the greedy direction, a major drawback is the difficulty of easily incorporate constraints. Also, a thorough and technical study must be done to break the complexity of the proposed learning algorithm. Works tailored in this approach include Bennett and Blue (1996); Tjortjis and Keane (2002); Garofalakis *et al.* (2003); Fromont *et al.* (2007); Kocev *et al.* (2007); Auer *et al.* (1995); Tzirakis and Tjortjis (2017), Turney (1995); Estruch *et al.* (2002); Esmeir and Markovitch (2004, 2006); Madzarov *et al.* (2009); Omielan and Vadera (2012); Krętowski and Grześ (2006) and Gehrke *et al.* (1999). Works using this approach (Fromont *et al.*, 2007; Nijssen and Fromont, 2007) can sometimes guarantee the optimality of decision trees with restrictive assumptions on the search space (for instance the binarity of the tree, the number of constraints, the size of the dataset).

3.4.3 Linear, SAT and Constraint Programming Approaches

The discrete nature of the search space and the logical constraints that can be imposed on the decision trees are highly related to constraint satisfaction problem (CSP) or linear programming problems or SAT as well. In contrast to top-down greedy and safe enumeration methods, these approaches formalise the problems in clear mathematical form (*e.g.*, linear programming, SAT, MaxSAT or CP) so that it can be optimised by an appropriate solver.

At first glance, this approach would provide optimal guarantees. In fact, formulating the tree learning problem in terms of global optimisation makes it possible to focus more on the modelling than on the algorithm. This is very suitable for providing mathematical guarantees such as fairness (Aghaei *et al.*, 2019). Also, it allows an easy way to integrate constraints as regularisation terms, even for structure-level constraints (Bertsimas and Dunn, 2017). However, optimality is not usually reached due to the large computational time required. Therefore, most works require acceptable initial solutions to speed up the optimisation. Furthermore, the majority of CP, ILP and MIP problems are NP-complete, even with restrictive assumptions. Thus, the scalability of the problem and the sub-optimal solutions a given time limit remain problematic. Related works include Heidenberger (1996); Bessiere *et al.* (2009); Verwer and Zhang (2017); Bertsimas and Dunn (2017); Menickelly *et al.* (2016); Narodytska *et al.* (2018); Firat *et al.* (2020); Verwer and Zhang (2019); Aghaei *et al.* (2019) and Bennett and Blue (1996).

3.4.4 Probabilistic Approach

Bayesian methods give a probabilistic formulation of the problem. They are efficient to integrate several constraints with priors. Few works on this approach exist because the choice of the prior and computing the posterior are still open problems. Works that propose priors and approximations of the posterior include Chipman *et al.* (1998); Denison *et al.* (1998); Buntine (1992); Schetinin *et al.* (2007); Nijssen (2008); Nuti *et al.* (2019). The Bayesian formulation has the advantage to integrate constraints with a clear and rigorous mathematical way through priors.

Having the idea that a decision tree can be framed into a probabilistic formulation, works have tried to transform the non-parametric problem into a parametric one, and the combinatorial space into a real space approximation, to learn trees with standard gradient descent optimisation. However, currently, works that succeed with this approach only learn multivariate decision tree under structure-level constraints (e.g, Norouzi *et al.* (2015)).

3.4.5 Summary About Categorisation of Approaches

To summarise, we have identified several optimisation methods which we categorised in top-down greedy induction, safe enumeration approaches, mixed integer programming, probabilitic approaches (see Table 3.1 for a global summary). Due to their low computational cost, historical developments in early methods such as CART and C4.5 and their ease of implementation and modification, greedy top-down methods are the most studied. The underlying top-down induction algorithm remains similar in structure and is easy to understand, yet it may provide sub-optimal solutions due to its greedy nature. On the other hand, safe enumeration methods are computationally costly but are likely to produce more accurate decision trees. However, finding a solution with safe enumeration may be difficult because different constraints have to be simultaneously satisfied, making the search more complex. Bayesian approaches are challenging because priors must be carefully designed to soundly enforce constraints. They are also rather expensive due to the use of sampling strategies. Finally, CP/SAT/MIP models offer an alternative to mathematically specify constraints. If decision trees are part of a bigger problem of a decision system that need to impose some constraints, these additional constraints only need to be formulated in the SAT/CP/MIP modelling framework in order to learn trees that comply with those constraints by design. However, these CP/SAT/MIP methods are expensive to use and often require to be initialised with satisfactory solutions.

In summary, this survey shows that there is no one-size-fits-all solution to the problem of learning decision trees with user and domain knowledge constraints. Depending on the characteristics of the dataset itself, the type and number of constraints, the acceptable discrepancy in accuracy and the available computational time, one method will be preferred to the others. The high computational cost of non-greedy methods explains their infrequent use (until recently).

Table 3.1 confirms that top-down greedy algorithms have been largely studied. Many works have proposed to improve them, which is natural since they were prominent in both the literature and practical applications and they were easier to modify and to implement than their computational costly competitors. This has to be put in perspective of recent works that are focusing on linear and constraint programming approaches thanks to efficient implementation supported by faster computations. This trend is likely to only increase in the future, leading to new developments.

3.5 Discussion

This survey reviewed how constraints can be defined and applied to decision trees, to make them safer, more accurate, more understandable, more robust, or more trustworthy. This section provides a discussion. Specifically, we express the difficulty of traditional algorithms to enforce constraints on decision trees. The optimality and interpretability of learned decision trees under constraints are also discussed. Finally, we mention what could be future research directions in this field. Note that this section has a marginal modification from the discussion provided in **Nanfack** *et al.* (2022a).

3.5.1 On the Weaknesses of Standard Top-down Induction Algorithms to Constrain Decision Trees

First of all, it is important to note that pruning techniques may fail to learn more interpretable decision trees, even though they can reduce the complexity of the tree. Indeed, pruning methods do not discover richer trees than traditional top-down greedy algorithms. In other words, pruned greedy top-down trees may sometimes appear as *unnatural* and it is possible to find smaller and better (in terms of accuracy and interpretability) trees, even manually (Piltaver *et al.*, 2016). Secondly, according to what has been shown in Section 3.3, with traditional algorithms, there is a need to adapt each constraint to a new specific heuristic: for example, ranked version of information gain (Hu *et al.*, 2010, 2012) for monotonicity constraint, information gain sensitivity for fairness (Kamiran *et al.*, 2010) and so on. This is relatively inefficient when many constraints or properties must be guaranteed. Indeed, the problem becomes much more complex and local heuristics are susceptible to stuck in local solutions. There, further studies are required to propose flexible impurity measures that can integrate constraints more easily.

Table 3.2: A benchmark for depth-constrained decision tree learners. Maximum depth is set to 3. Train and test columns indicate respectively the mean of training and testing accuracy over 5 independent runs whereas ctime columns refer to the computational time (in seconds). CART and BinOCT are run with continuous features while others require binarisation of continuous features. Table taken from **Nanfack** *et al.* (2022a)

	Top-down greedy			MIP						SAT			CP			Safe enumeration					
	CART			BinOCT			OST			MaxSAT_DT			Verhaeghe *et al.* (2019)			DL8.5			OSDT		
	Breiman *et al.* (1984)			Verwer and Zhang (2019)			Aghaei *et al.* (2021)			Hu *et al.* (2020)						Aglin *et al.* (2020)			Hu *et al.* (2019)		
Dataset	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime
Balance S.	70.51	66.24	0.0	74.36	68.79	600.0	–	–	–	–	–	–	–	–	–	75.00	**69.94**	0.0	–	–	–
Banknote A.	94.38	93.82	0.0	97.47	**96.15**	600.0	93.25	93.08	36.0	93.37	92.54	18.0	93.37	92.54	0.0	93.37	92.54	0.0	93.37	92.59	14.0
Biodeg	83.92	78.03	0.0	82.76	79.85	600.0	79.58	76.04	604.0	82.02	79.17	599.0	83.54	80.08	2.0	83.54	80.08	1.0	82.23	**80.53**	128.0
Car	80.61	**80.40**	0.0	80.94	79.21	600.0	–	–	–	–	–	–	–	–	–	81.57	79.82	0.0	–	–	–
Credit A.	87.28	84.89	0.0	88.63	85.24	600.0	86.71	85.67	602.0	89.20	85.49	598.0	89.61	**86.10**	1.0	89.57	85.74	0.0	86.71	85.37	0.0
Hepatitis	91.48	**83.76**	0.0	92.59	80.51	600.0	82.54	82.69	521.0	91.03	81.54	599.0	91.55	82.05	1.0	91.38	81.54	0.0	79.31	79.49	0.0
Ionosphere	93.03	88.89	0.0	93.00	87.95	600.0	87.55	83.24	602.0	91.56	86.82	598.0	93.16	**89.32**	6.0	93.08	88.64	2.0	82.66	77.73	29.0
Iris	97.22	**95.61**	0.0	92.86	88.95	110.0	–	–	–	–	–	–	–	–	–	98.22	93.68	1.0	–	–	–
Mammo. M.	84.87	83.23	0.0	85.34	**83.85**	600.0	84.45	83.65	602.0	85.31	83.65	599.0	85.31	83.56	1.0	85.27	83.46	0.0	84.21	82.79	9.0
Monk1	78.18	81.14	0.0	90.22	**86.33**	524.0	90.41	85.07	186.0	90.46	85.32	5.0	90.46	85.18	0.0	90.46	85.18	0.0	90.46	85.18	128.0
Monk2	66.12	63.50	0.0	68.04	59.34	600.0	65.78	**65.56**	601.0	68.27	59.34	600.0	68.62	57.75	0.0	68.40	58.67	0.0	65.78	**65.56**	0.0
Monk3	98.85	**99.12**	0.0	98.89	98.99	83.0	98.23	97.60	27.0	98.89	98.99	1.0	98.89	98.99	0.0	98.89	98.99	0.0	97.16	96.12	0.0
Pima I. D.	79.21	72.69	0.0	80.45	73.44	600.0	77.03	74.13	601.0	78.06	71.56	599.0	78.33	70.52	0.0	78.33	70.52	0.0	76.70	**74.58**	128.0
Post O. P.	76.24	71.21	0.0	81.54	**72.73**	600.0	–	–	–	–	–	–	–	–	–	84.92	66.36	0.0	–	–	–
Seismic	93.52	93.14	0.0	93.73	93.16	600.0	93.41	**93.45**	604.0	93.47	93.22	600.0	93.47	93.28	0.0	93.47	93.19	0.0	93.42	93.44	0.0
Spambase	89.04	**87.63**	0.0	85.54	85.09	601.0	83.76	83.67	607.0	83.77	83.30	600.0	84.22	83.75	0.0	84.22	83.76	0.0	84.22	83.84	128.0
Spect H.	81.33	75.46	0.0	82.10	77.91	600.0	81.67	75.62	29.0	82.20	75.52	255.0	82.20	**79.10**	0.0	82.20	**79.10**	0.0	80.80	77.01	128.0
Thoracy S.	87.25	83.24	0.0	87.61	82.20	600.0	85.23	**84.75**	601.0	87.90	81.86	599.0	88.12	81.86	1.0	88.01	80.85	0.0	85.23	**84.75**	0.0
Tic T. T.	75.91	72.92	0.0	77.19	71.67	600.0	75.49	73.06	602.0	76.99	73.92	599.0	78.80	73.17	1.0	78.50	73.33	0.0	77.52	**74.17**	128.0
Wine	99.33	**93.09**	0.0	99.85	91.56	381.0	–	–	–	–	–	–	–	–	–	97.89	92.44	0.0	–	–	–

3.5.2 On the Optimality of the Learned Decision Trees under Constraints

Before discussing optimality of existing techniques, it may also be relevant to look back at their history. Regarding the techniques proposed by the literature and by looking at Table 3.1, we can note that the earliest methods (Quinlan and Rivest, 1989; Núñez, 1991; Garofalakis *et al.*, 2000; Potharst and Feelders, 2002) with constraint enforcement are top-down. A reason could be the fact that this period was extremely dominated by the heuristic search for combinatorial problems. Bayesian (Chipman *et al.*, 1998; Denison *et al.*, 1998; Buntine, 1992) and MIP methods (Heidenberger, 1996) also appeared early. Due to their computational cost, these methods have been less used than greedy methods. However, nowadays, the computing capabilities of machines and the efficiency of solvers have highly increased and methods that were deemed as inefficient because of their computation cost are now re-emerging. This is the case for MIP and Bayesian methods. By nature, greedy algorithms can rarely lead to optimal decision trees given a particular set of constraints. If correctly modelled, MIP/CP/SAT approaches produce optimal decision trees given enough time. However, from what have been seen for current methods, there is still room for improvement in the modelling, since the majority of current CP/SAT methods require binary features and are limited to binary classification whereas current MIP methods (such as Verwer and Zhang (2019); Bertsimas and Dunn (2017)) make use of starting solutions to speed up the search.

Table 3.3: A benchmark for depth-constrained decision tree learners. Maximum depth is set to 4. Train and test columns indicate respectively the mean of training and testing accuracy over 5 independent runs whereas ctime columns refer to the computational time (in seconds). CART and BinOCT are run with continuous features while others require binarisation of continuous features. Table taken from **Nanfack** *et al.* (2022a)

	Top-down greedy			MIP						SAT			CP			Safe enumeration					
	CART			BinOCT			OST			MaxSAT_DT			Verhaeghe *et al.* (2019)			DL8.5			OSDT		
	(Breiman *et al.*, 1984)			(Verwer and Zhang, 2019)			(Aghaei *et al.*, 2021)			(Hu *et al.*, 2020)						(Aglin *et al.*, 2020)			(Hu *et al.*, 2019)		
Dataset	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime	train	test	ctime
Balance S.	72.86	67.07	0.0	77.39	70.83	600.0	–	–	–	–	–	–	–	–	–	79.06	**72.49**	0.0	–	–	–
Banknote A.	97.24	95.92	0.0	97.71	**96.73**	600.0	93.37	93.29	603.0	94.81	94.40	599.0	94.83	94.52	1.0	94.83	94.58	0.0	93.99	93.24	10.0
Biodeg	87.16	78.07	0.0	82.17	78.33	601.0	81.07	77.08	608.0	84.02	79.17	598.0	87.03	**80.45**	171.0	86.78	80.38	26.0	82.28	78.79	10.0
Car	81.23	80.38	0.0	83.02	82.18	600.0	–	–	–	–	–	–	–	–	–	84.55	**82.82**	0.0	–	–	–
Credit A.	90.14	85.03	0.0	89.12	84.88	600.0	87.58	85.67	604.0	89.98	85.00	598.0	91.86	85.12	42.0	91.74	85.25	6.0	89.86	**85.73**	10.0
Hepatitis	94.54	84.33	0.0	96.90	79.49	600.0	82.54	**85.26**	601.0	95.86	78.97	598.0	97.76	78.46	4.0	97.76	78.46	1.0	79.31	79.49	0.0
Ionosphere	95.31	88.89	0.0	94.75	**89.32**	600.0	89.45	85.23	604.0	92.24	85.91	597.0	97.26	84.32	549.0	97.26	84.32	44.0	82.66	77.73	3.0
Iris	99.01	**95.61**	0.0	100.00	94.74	9.0	–	–	–	–	–	–	–	–	–	98.39	93.68	0.0	–	–	–
Mammo. M.	85.41	82.80	0.0	85.98	82.60	600.0	84.53	**83.65**	603.0	85.88	82.98	598.0	86.43	82.60	2.0	86.40	82.60	0.0	84.50	82.60	10.0
Monk1	84.01	83.29	0.0	100.00	**100.00**	62.0	100.00	**100.00**	169.0	100.00	**100.00**	0.0	100.00	**100.00**	0.0	100.00	**100.00**	0.0	100.00	**100.00**	0.0
Monk2	68.59	64.97	0.0	71.51	57.35	600.0	65.78	**65.56**	602.0	69.69	61.72	599.0	72.71	57.75	1.0	72.58	58.15	0.0	65.78	**65.56**	0.0
Monk3	98.85	**99.12**	0.0	98.89	98.99	600.0	98.37	98.02	64.0	98.94	98.56	488.0	98.94	98.56	1.0	98.94	98.56	0.0	97.16	96.12	0.0
Pima I. D.	81.48	71.47	0.0	79.97	71.77	600.0	76.87	74.09	603.0	78.92	71.46	599.0	81.11	69.27	5.0	80.97	70.52	1.0	76.70	**74.58**	10.0
Post O. P.	79.32	**70.20**	0.0	85.23	64.55	600.0	–	–	–	–	–	–	–	–	–	91.69	65.45	0.0	–	–	–
Seismic	93.95	92.71	0.0	93.78	93.25	600.0	93.41	**93.46**	608.0	93.55	93.07	599.0	93.57	93.13	2.0	93.57	93.06	0.0	93.42	93.44	0.0
Spambase	91.16	**89.75**	0.0	81.50	81.48	603.0	84.58	83.91	613.0	84.28	83.20	599.0	85.50	84.40	2.0	85.50	84.40	0.0	81.83	81.22	10.0
Spect H.	83.78	77.11	0.0	85.70	**79.40**	600.0	81.12	76.49	195.0	86.20	77.01	599.0	86.90	77.01	3.0	86.90	77.01	0.0	84.20	76.72	10.0
Thoracy S.	88.45	82.86	0.0	88.69	80.68	600.0	85.23	**84.75**	602.0	89.15	81.02	598.0	90.28	80.68	8.0	90.17	80.17	1.0	85.23	**84.75**	0.0
Tic T. T.	83.70	**82.31**	0.0	82.28	79.00	600.0	80.22	76.98	604.0	81.50	76.75	598.0	87.05	80.83	7.0	86.69	81.25	1.0	81.62	78.17	10.0
Wine	100.00	**92.10**	0.0	100.00	88.89	178.0	–	–	–	–	–	–	–	–	–	100.00	89.33	0.0	–	–	–

Added with the scalability problem, these issues remain current limitations of MIP/CP/SAT approaches.

Table 3.2 and 3.3 show a benchmark of extensive experiments that we performed on current state-of-sthe-art depth-constrained[2] decision tree methods. We reported training, testing (75-25 train-test split percentage) and computational time over five independent runs (with 10 minutes as time limit for each run to prevent unlimited computations, in accordance with Verwer and Zhang (2019)). Cells of the table marked with '−−' correspond to methods that do not work with multi-class classification. Methods are evaluated on well-known datasets (listed on the first column of the tables) from the UCI (Dua and Graff, 2017) repository. Except CART and BinOCT (which are used with continuous features), all these methods have the strong requirement to work only with binary features, thus continuous features have to be discretised[3] (Verhaeghe *et al.*, 2019; Hu *et al.*, 2020). It can be seen that the old top-down greedy CART method is still competitive in terms of predictive accuracy over MIP/SAT/CP and safe enumeration methods. However, regarding the training performance, it usually fails to provide optimal or near-to-optimal solutions, unlike MIP/SAT/CP and safe enumeration methods. It is also surprising that the performance improvement of MIP/SAT/CP and safe enumeration methods do not necessarily lead to good generalisation yet having a depth constraint, which is supposed to reduce the generalisation gap. This suggest

[2]Without a constraint on the min/max number of instances on leaves

[3]We used the KBinsDiscretizer from the Scikit-learn library, with 3 Bins.

Table 3.4: Methods, their number of variables and their number of constraints. N is the number of instances of the dataset, M is the number of features, T_{all} is the total number of splits and $T_{\max}$ is the number of maximum split per feature in Verwer and Zhang (2019). S is the number of nodes or the size of the tree, K is its depth, L is its number of leaves and C is the number of classes. Only the work in the last row tries to incorporate a novel constraints while the rest of works aim to make accelerate the learning.

Methods	**Number of variables/literals**	**Number of constraints/clauses**
Bessiere *et al.* (2009)	$\mathcal{O}(S \times M)$	$\mathcal{O}(M \times S^2 \times N^2 + S \times M^2 + M \times S^3)$
Narodytska *et al.* (2018)	$\mathcal{O}(S^2 + M \times S)$	$\mathcal{O}(M \times S^2 + M \times N \times S)$
Avellaneda (2020)	$\mathcal{O}(2^K(M + N + C))$	$\mathcal{O}(2^K(M^2 + N \times M + C)$
Bertsimas and Dunn (2017)	$\mathcal{O}(2^K(M + N + C))$	$\mathcal{O}(2^K(C + N \times K + M))$
Verwer and Zhang (2019)	$\mathcal{O}(2^K(M + C + \log(T_{\max}))$	$\mathcal{O}(N + 2^K(M \times T_{\text{all}} + C))$
Aghaei *et al.* (2021)	$\mathcal{O}(2^K(N + M))$	$\mathcal{O}(2^K(N + M))$
Aghaei *et al.* (2019)	$\mathcal{O}(L(M \times S + N))$	$\mathcal{O}(L(L \times N + M \times S))$

that more inductive biases should be proposed on these methods in order to further reduce this generalisation gap.

Regarding the computational time, it appears from the table that MIP and MaxSAT require more time to find (optimal) solutions (or at least to prove optimality) compared to CP, safe enumeration and greedy methods such as CART. Indeed, CART uses a heuristic and its implementation in Scikit-learn is highly optimised with system calls using the C language. Regarding safe enumeration methods, they use highly optimised C libraries too. Therefore, it remains unclear whether this improvement of computational time comes from this use of C libraries or from a reduction of theoretical complexity. Therefore, we highlight that future work should be done in order to fairly analyse computational time in the lens of theoretical complexity evaluation, which is currently lacking in some extent.

3.5.3 On the Complexity of SAT/CP/MIP Formulations of the Optimal Decision Trees

Although being theoretically hard (NP-hard), the optimal decision tree problem under structure-level constraints is attracting researchers. Indeed, several works have been proposed as explained in Section 3.3.1 and work well in practice with a limited budget of computational time (usually minutes or hours).

Table 3.4 compares several of these methods in terms of their number of variables/literals and number of constraints/clauses (that we have counted if they were not mentioned in the paper). The number of variables and number of constraints serve as a heuristic to assess which method is more attractive in terms of practical computational time and therefore time and space complexity. Among others, the formulation of Aghaei *et al.* (2019) on the last row of the table is the only method, among SAT/CP/MIP formulations, whose goal is not to speed up computational time, but rather to leverage global optimisation to enforce fairness constraints. All other methods usually aim to speed up optimisation when learning optimal decision trees under structure-level constraints. The first three works are SAT-based methods. From the first row to the third, the number of literals or constraints is enhanced in order to improve computational time. This can also be noted for other methods. For example, the BinOCT formulation of Verwer and Zhang (2017) is an improved MIP version of the OCT formulation (Bertsimas and Dunn, 2017) where the number of variables does not depend on the dataset size N.

As mentioned above, the number of variables and constraints are used here as simple heuristics to characterise the difficulty and scalability of a particular method. Therefore, future studies should be done in order to access whether this reduction in the number of variables and constraints of these methods leads to an improved computational complexity. This issue is also an open problem for safe enumeration methods. Among these methods only Auer *et al.* (1995) accompanied their T2 algorithm with a polynomial time complexity for the class of decision tree of depth at most 2. Hence, in the future, newly proposed methods should also discuss their computational complexity rather than just comparing number of variables/constraints or reporting only the empirical computational time.

3.5.4 On the Interpretability, Trustworthiness and Robustness of Decision Trees

According to what has been related previously in the literature (*e.g*, Verwer and Zhang (2017); Bertsimas and Dunn (2017) in Section 3.3.1), the question of interpretability is generally related to the complexity of the decision tree. Thus, when authors talk about forcing trees to be more interpretable and easier to understand, they commonly think about reducing complexity, *i.e.*, structure-level constraints (Verwer and Zhang, 2017; Bertsimas and Dunn, 2017; Verwer and Zhang, 2019). However, decision trees that are learned with a maximal depth of two can be too simple in certain contexts, such as in the medical domain. That is to say, domain experts or users will not trust the learned tree since its decision rule is too simple (Freitas, 2014). Therefore, this conducts to a loss of interpretability because the domain expert will consider "an incompleteness criterion of models" (Guidotti *et al.*, 2018). Thus, similarly to the fact that increasing the size, depth and number of leaf nodes of decision trees may lead to a loss of interpretability, decision trees with a very small size and depth, even accurate, are likely to not be accepted because of their "over-simplistic explanations" (Freitas, 2014).

Hence, trustworthy decision trees with domain knowledge constraints and user-defined constraints are needed to increase the level of interpretability of the decision tree. Despite the work of Núñez (1991); López-Vallverdú *et al.* (2007, 2012) to learn more comprehensible and trustworthy decision trees, this direction is not yet sufficiently studied in the literature. In fact, further away than domain knowledge constraints, trustworthy decision trees also include decision trees with ethical guarantees like the fairness constraint. More work needs to be done on that topic. Furthermore, beyond complexity, there is a large gap in the general direction of the possibility to impose multiple constraints, although Aghaei *et al.* (2019) recently showed how to learn (optimal) decision tree under fairness constraints.

Robustness is another particular issue in machine learning and thus in decision trees. Recent challenges in machine learning have shown a stream of interest in making machine learning classifiers robust to adversarial examples. Although this is very frequent in deep learning, it has also been shown that those adversarial examples can be transferred to any classifier and thus to a decision tree (Papernot *et al.*, 2016). Furthermore, because adversarial examples can be generated via constraints (Kantchelian *et al.*, 2016; Biggio and Roli, 2018), improving the robustness of classifiers can also be done via constraint enforcement (Bastani *et al.*, 2016). And yet, there are only very few works (Chen *et al.*, 2019), on constraining the decision tree to be more robust to particular attacks of adversarial examples.

3.5.5 On the Usefulness of Constraint Enforcement for Approximating Black-box Machine Learning Algorithms

Several works presented in Section 3.3.1.1 get benefit from constraint enforcement to approximate and explain a black-box machine learning algorithm. In fact, explaining black-box machine learning is (obviously) necessary from an ethical perspective and for preventing a "black-box society" (Pasquale, 2015; Guidotti *et al.*, 2018) guided by sense(less) decisions of algorithms. On the other hand, it would be useless to learn a decision tree that is difficult to understand if its purpose is to explain a black-box model. Works like Craven and Shavlik (1995); Boz (2002); Zilke *et al.* (2016); Yang *et al.* (2018) have been proposed to explain black-box models. They can guarantee clear explanations by constraining the tree to be small and less deep. Of course, constraining the size of trees can affect the level of comprehensibility of the decision tree for experts helping them to understand how the initial model works. However, restraining too much the size of the trees may have consequences on their prediction performances thus being so different from the initial model that they would become useless. It is, therefore, necessary to find a good balance between the tasks of providing clear explanations and getting closer to the initial model. In this direction, there is still a significant gap in the literature to provide theoretical guarantees on the fidelity of the interpretation of the model explanations.

3.5.6 Future Prospects

In addition to previously mentioned gaps and research directions that need to be explored, this section presents other relevant future directions where research should be conducted.

3.5.6.1 Tree Balance Constraint, Interpretability of Decision Trees

Regarding structure-level constraints, several open issues are identified. The first issue is related to the capability to impose the balance constraint on the structure of the decision tree. The second one targets the trade-off between the small size, the depth or the number of leaf nodes in the tree and the question of the degree of interpretability of the decision tree. In fact, forcing the smallness/sparsity of a tree can improve its interpretability, but when it becomes too small, even remaining very accurate, its interpretability can decrease, since learned rules may become unreliable w.r.t. the domain expertise. Third, it is important to look for flexible optimal decision tree formulations with fewer restrictions (categorical variables as well as real variables, binary as well as non-binary trees, classification as well as regression trees) and that require less computation time. Therefore, a good modelling for decision trees should be much more flexible in terms of binarity, type of variables, tasks, depth, size, number of leaf nodes in order to be closer to the specifications of the user and the domain experts.

3.5.6.2 Experimental Settings

Regarding attribute-level and instance-level constraints, experimental evaluations are relatively limited because there is not enough datasets that provide domain knowledge as constraints, except for attribute costs (mentioned in Section 3.3.2.2). Hence, more datasets should be proposed to benchmark algorithms.

3.5.6.3 Flexible Impurity Measures

In real-world applications, top-down greedy algorithms are frequently used but we have previously mentioned the limitations of such approaches (see Section 3.4.1). In fact, the well-known impurity measures that are based on entropy and the Gini index do not easily integrate constraints. Even if pruning methods attempt to make the learned tree satistying the constraints, the final tree may seriously lose its performance. Therefore, more flexible heuristics that make it easier to incorporate constraints should be proposed. Furthermore, the theoretical limits of newly introduced measures should be studied.

3.5.6.4 Domain Knowledge Constraints (feature and instance-level) for Constraint Programming and Bayesian Formulations

Table 3.1 shows that few works have used probabilistic formulations to enforce attribute and instance-level constraints, which is still an open issue. Yet, Bayesian formulations have the advantage of learning decision trees with a global objective or, in certain cases, a global heuristic. This would allow to soundly express global constraints such as instance-level constraints. In Bayesian methods, one can enforce constraints in a clear mathematical way through priors. In this direction, Angelopoulos and Cussens (2005a) use informative priors to allow “box” constraints (which can be seen as rules constraints in attribute-level). Their formulation performs well on a Bayesian predictive model with an ensemble of trees, not for a single tree. However, Nijssen (2008) makes it possible to learn a single accurate tree with his Bayesian formulation, but without investigating how to integrate informative priors. Further studies should be done in order to allow informative priors with domain knowledge constraints, such as rules in the Bayesian formulations, in order to learn a single accurate and trustworthy decision tree.

Beyond Bayesian formulations, linear and constraint programming formulations are also used to learn decision trees with a global objective. Thus, this approach is well suited to integrate attribute and instance-level constraints at a global level (for instance attribute costs, hierarchy, fairness). Such constraints are not trivial to implement with *e.g.*, greedy methods (Struyf and Džeroski, 2006). However, the majority of the proposed formulations have focused their attention on enforcing structure-level constraints, such as imposing the tree to be small or shallow. Thus, more efforts should be done to propose flexible linear programming approaches, which can integrate various types of constraints, including hierarchy and rule constraints that are important for the trustworthiness of decision trees for critical domains.

3.5.6.5 Constraints for Proxy Models

In the motivational part of the survey (Section 3.1), we emphasised the importance and necessity to impose constraints on decision trees used as a proxy for black-box models. As a reminder, a proxy (or surrogate) decision tree that approximates and explains a black-box model can be too small and not being able to approximate the black-box model; or it can be so deep that it is not able to provide human-understandable explanations. According to the literature that has been examined throughout this chapter, it is clear that it is still an open issue for future research because the learned trees might be unreliable and untrustworthy. Future works on decision trees as proxy models should get benefit from constraint enforcement on decision trees in general, to ensure that the approximating tree of a black-box model meets the same guarantees as the approximated model. Moreover, further studies should also study whether constraints satisfied by the black-box model such as fairness constraints can be transferred to proxy decision trees.

3.6 Conclusion

This chapter reviewed current and past literature on works that learn constrained decision trees. We begin by clarifying the importance of constraint enforcement on decision trees, for instance to meet a specified level of interpretability or trustworthiness. We also present a taxonomy which comprises structure-level constraints, attribute-level constraints and instance-level constraints. Our findings reveal that a large part of methods which enforce structure-level constraints (i.e., through the number of leaf nodes, depth and size) aim to improve the accuracy and interpretability of decision trees. The two other levels of constraints usually aim to guarantee that decision trees comply with the requirements of a particular domain, often provided by domain experts. For example, it may be necessary to enforce monotonicity for the predictions with respect to some attributes. This makes decision trees more reliable and predictions appear as more realistic and similar to those that would be made by humans in the same context.

Historically, top-down greedy algorithms such as CART and C4.5 have been prominent in early developments for decision tree induction. They are therefore widely used to learn decision trees under constraints. Top-down greedy induction approaches reported in this work enforce constraints by applying pruning methods or with specifically modified heuristics. Despite being the most popular approach, top-down greedy induction methods usually produce sub-optimal solutions for the training performance, which is not the case of other approaches that we have identified. Linear programming, constraint programming and Bayesian formulations provide optimal solutions when they are given enough time to explore solutions but they are more computationally demanding. With recent improvements in implementations and computational power, these methods are gaining interest. Despite being able to reach top training performance, by experimentally benchmarking depth-constrained state-of-the-art tree learners, we showed that this training performance is not usually translated to predictive performance.

We suggest that further research should be intensified in the following areas. First, providing humanly understandable explanations of black-box machine learning models should be performed with theoretical guarantees (regarding both the predictions and the constraints already satisfied in the black-box model). Second, theoretical and technical limitations of works that learn structure-level constrained and optimal decision trees should be studied. Third, research is also needed to study possible compatibility between different types of constraints. Lastly, our study encourages the scientific community to propose more datasets with domain knowledge in order to systematically validate methods without the need of experts.

Chapter 4

Constraint Enforcement on Decision Trees Using Top-down Greedy Methods: Case of Fairness Constraints

This chapter presents a boundary-based method to learn decision trees under the fairness constraints. The chapter is based on our publication entitled "Boundary-Based Fairness Constraints in Decision Trees and Random Forests" (**Nanfack** *et al.*, 2021a). Here, we will mainly focus on decision trees, though in the paper we also showed results on tree-based ensemble methods such as random forests.

Contents

4.1 On the Need for Fair Learning Algorithms

In the introductory chapter 1 of this thesis, we highlighted the fact that machine learning is increasingly applied in various domains that include sensible and high-stake domains such as finance, criminal justice, human resource management (job hiring) and health. However, when applied on these sensitive domains where observations (data sample) are humans, there is a risk that algorithmic decisions provided by machine learning models violate laws regarding human rights. Indeed, these algorithmic decisions may disproportionally benefit to a certain group of individuals, making the decisions discriminatory and therefore violate anti-discrimination laws. An example of such laws is the US “80 percent rule” (disparate impact, see Section 4.1.2) law advocated by the Equal Employment Opportunity Commission (EEOC), which requires that the acceptance rate for any race, sex, or ethnic group should be at least 80% of the one for the group with the highest rate (Feldman *et al.*, 2015; Pessach and Shmueli, 2022). Hence, violations of such kind of non-discrimination laws by machine learning models make their decisions discriminatory, *unfair* and illegal. If nothing is done to overcome the risk of unfair decisions, this concern may seriously hinder the deployment of machine learning models in various sensitive applications.

4.1.1 On the Source of Unfairness in Machine Learning Models

To overcome the possible risk of unfair decisions by models, it is important to state the origin of potentially unfair models. As any learned model comes from a learning algorithm, which takes as input data and since this learning algorithm is designed by a machine learning practitioner, most sources of potential learned unfair decisions can be divided into three categories: unfairness from data, unfairness from the learning algorithm and unfairness from user interactions. These unfairness sources can be analysed with the lens of statistical bias. A thorough discussion of different sources can be found in Pessach and Shmueli (2022). In the following, we briefly summarise these three important sources.

4.1.1.1 Unfairness from Data

The most common (data-related) causes that lead to unfair machine learning models are the representational and sampling (selection) biases.

The *representation bias arises from how we sample from a population during the data collection process* (Suresh and Guttag, 2019; Pessach and Shmueli, 2022). This is the case where samples lack diversity of the population. For example, well-known datasets such as ImageNet (Russakovsky *et al.*, 2015) lack a geographical diversity and exhibits a bias towards Western cultures (Pessach and Shmueli, 2022).

Also related to the representation bias, the sampling (selection) bias arises due to non-random sampling of the population. Technically, it introduces a bias in $p(\mathbf{X})$, where $\mathbf{X}$ is a sample. One example provided by Zadrozny (2004) is the case where the probability of selecting one example depends on some features $\mathbf{x}_j$, but is independent of the label random variable y given $\mathbf{x}$. Hence, the selection is not only *completely at random*, i.e., the selection is biased [1] and also $p(s|\mathbf{x}, y) = p(s|\mathbf{x})$, where s is the binary random variable controlling the selection of examples. A concrete example is provided by Suresh and Guttag (2019) where the target population for a disease may be all adults. However, we may only have medical records for the sample of people who were considered serious enough such that they had to go to the hospital and perform further tests. As a result, a model trained on this sample will not reflect the trend of the target population.

One can argue that discriminative classifiers (which do not make any assumption regarding $p(\mathbf{X})$ or the data generating process) do not suffer from the sampling selection bias. However, in practice, classifiers do suffer, especially when estimating their parameters (e.g., optimising the splitting rules for nodes in decision tree learners) and most importantly when evaluating their performance.

All these sources of unfairness may result in unfair and biased models. Although being sometimes subject to debate, the term "data biases" usually, refers to all processes that introduced a bias before "the data to the learning algorithm" step.

4.1.1.2 Unfairness from the Learning Algorithm

Machine learning models and their predictions are also the outcome of a learning algorithm. Therefore, even when there are no sampling or representation bias, the learning algorithm may produce models whose predictions favour certain groups of

[1] The unbiased selection is defined for the case where the selection s is independent of the example $\mathbf{x}$ and independent of the label y according to Zadrozny (2004).

individuals. Although, there is refrain [2] that the unfairness and bias of models are purely because of data-related biases, as stated by Suresh and Guttag (2019), data should not only be blamed. Indeed, when fitting models through optimisation in learning algorithms, we often lack a mechanism of control over produced models. This is because, the optimisation problem may be ill-defined (e.g., having multiple solutions, not playing its intended role) or may simply produce local optimal solutions, etc. There is therefore little to no guarantee that the learning algorithm may not produce a model whose decisions favour certain groups of the population.

4.1.1.3 Unfairness from the User Interaction

The same previously reasons invoked (for the unfairness from the learning algorithm) apply to the unfairness from the user interaction (here a machine learning practitioner). Indeed, a model is also the result of a machine learning practitioner, who makes choices regarding the model (hypothesis) class, the objective function, the hyper-parameters tuning, etc. Even with the absence of data-related biases, these choices are susceptible to introduce certain biases, and therefore may lead to models whose predictions favour particular groups of individuals.

In summary, one can see that of unfairness in machine learning can come from each component included in a classical machine learning "pipeline". Although appearing somewhat intuitive, the (un)fairness notion in machine learning still need to be technically defined.

4.1.2 Fairness Notions in Machine Learning

Combatting fairness has an intensive history in philosophy and psychology (Pessach and Shmueli, 2022). Roughly speaking, fairness is the "*absence of any prejudice or favouritism towards an individual or a group based on their intrinsic or acquired traits in the context of decision-making*" (Saxena *et al.*, 2019; Pessach and Shmueli, 2022). There is a recent entire research line whose purpose is to define how to define and technically measure fairness in the context of machine learning. As stated in Chapter 3 of this thesis (see Section 3.3.2.5), this research has brought to light two principal types of unfairness: individual unfairness and group unfairness (Chouldechova and Roth, 2020; Zhang *et al.*, 2016). As also stated in Section 3.3.2.5, group fairness aims to make machine learning models treat (or predict) similarly members from different groups (according to a sensitive feature such as sex or gender). While not very studied, individual fairness pays more attention to a

[2]https://twitter.com/ylecun/status/1274782757907030016?s=20

similar treatment for two similar individuals from different groups.

As explained in Section 3.3.2.5, there are several statistical measures that were introduced particularly for group fairness. The most common studied are the demography (or statistical parity) and the disparate mistreatment (w.r.t. overall misclassification, true positive rate or equal opportunity, false negative rates). The extension for the equal opportunity is the equalised odds introduced by Hardt *et al.* (2016). Readers are referred to Section 3.3.2.5 for their technical definition.

4.1.3 Brief State of the Art on Mitigating Unfairness in Machine Learning

Several techniques have been proposed to mitigate unfairness of machine learning models. The categorisation usually employed in this literature use the terms: pre-processing, in-processing and post-processing.

Pre-processing techniques aim to transform the original training data $\mathcal{D}$ into $\mathcal{T}(\mathcal{D})$ such that when learning a model on this new training data, the learned model will not produce unfair predictions. For example, in Calmon *et al.* (2017), such a transformation is a probability distribution obtained through an optimisation process whose goal is to limit the dependence between the transformed outcome variable and the protected or sensitive feature.

Post-processing techniques aim to *debias* a learned model by usually considering it *agnostically* i.e., without assuming a model class. An example of such post-processing is the multiaccuracy boost introduced by Kim *et al.* (2019), where the goal is to *audit* the output model and thanks to sequential updates of the model, the accuracy of the classifier over protected groups is improved, thus improving fairness w.r.t. disparate mistreatment.

In-processing techniques are very popular since they target the learning algorithm. The goal is to change the objective function, for example, by reweighting training instances (Chai and Wang, 2022; Lahoti *et al.*, 2020) or by incorporating fairness constraints (Zafar *et al.*, 2017, 2019; Donini *et al.*, 2018). For example, the work of Zafar *et al.* (2017) on logistic regression and support vector machines (SVMs) tries to mitigate the unfairness through constraint enforcement. This work, which will be extended for decision trees in this chapter integrates the fairness constraint thanks to the following optimisation problem

$$\min_{\boldsymbol{\theta}} \ L(\boldsymbol{\theta}) \text{ s.t. either } \Big|\mathrm{Cov}\big(z, d_{\boldsymbol{\theta}}(\boldsymbol{x})\big)\Big| \leq \gamma \tag{4.1}$$

$$\text{or } \Big|\mathrm{Cov}\big(z, \min\big(0, y\, d_\theta(\boldsymbol{x})\big)\big)\Big| \leq \gamma,$$

where z is the sensitive (or protected feature such as sex, gender, race, etc.), $\boldsymbol{\theta}$ is the model parameters, $L(\boldsymbol{\theta})$ is the loss function (cross-entropy for the logistic regression and penalised hinge loss for SVMs), y is the class variable (having values in $\{-1, 1\}$, but easily extensible for multi-class classification), γ is the level of (proxy) unfairness, $d_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the signed distance [3] of the instance $\boldsymbol{x}$ to the decision boundary of the classifier. The first Cov term aims to (but not necessarily) push the independence between $d_\theta(.)$ and z while the second one aims to push (still not necessarily) the independence of this sensitive feature z to the degree of misclassification $\min(0, y d_{\boldsymbol{\theta}}(.))$. These terms are "proxy" measure to smoothly capture the unfairness. Note that the term "either" in the equation means that the unfairness measures are usually considered independently because of the *impossibility theorem*, which states that no more than one unfairness metric can hold at the same time for a well calibrated classifier (Kleinberg *et al.*, 2016; Saravanakumar, 2020).

4.2 Related Work on Fair Decision Tree Learners

Although intensively well studied for several differentiable models such as neural networks and logistic regression, there are not a lot of works that try to learn fairer non-differentiable models such as decision trees and ensemble of them. The few works that exist will be summarised in the following.

One of the first work on decision trees is the one of Calders *et al.* (2009). They propose data pre-processing by relabelling and reweighting instances so as to reduce bias in data. However, as pre-processing (i) cannot eliminate discrimination that may come from the learning algorithm, and (ii) may miss complex correlations with other features, the same authors later propose the first discrimination-aware tree learning algorithm (Kamiran *et al.*, 2010). They introduce a post pruning method consisting in leaf relabelling and the information gain sensitivity, which measures the level of discrimination induced by a split. Similarly, Raff *et al.* (2018) and Zhang and Ntoutsi (2019) develop a fair version of the impurity score. However, none of them are flexible, in the sense that they do not present a framework able, at the same time, to handle (i) different types of unfairness, (ii) multiclass classification and (iii) multiple sensitive features. Another notable work is the one of Aghaei *et al.* (2019) which proposes a mixed integer programming (MIP)

[3] The signed distance is positive if the instance is well classified and negative otherwise.

formulation to learn optimal fair decision trees. Nonetheless, MIP problems are computationally expensive.

Among the (i), (ii), (iii) mentioned desiderata, one of the methods that fulfills them is the proposal of Zafar *et al.* (2019) through Eq. 4.1. However, there are two challenges to adapt it for decision trees: (1) unlike logistic regression or SVMs which have, w.r.t. model parameters $\boldsymbol{\theta}$, an explicit formula for distance to the decision boundary ($\frac{\boldsymbol{\theta}^T\boldsymbol{x}}{||\boldsymbol{\theta}||}$ for linear SVMs and logistic regression), decision trees do not have such kind of formula, (2) optimisation of model parameters of standard decision tree learners such as CART cannot be framed as a differentiable and global optimisation such as in Eq. 4.1. The rest of this chapter copes with these challenges and allows to learn decision trees under boundary-based fairness constraints.

4.3 Distance to Decision Boundary for Decision Trees

In order to adapt the proposal of Zafar *et al.* (2019), one must first be able to compute the distance to decision boundary of instances given a decision tree.

By geometrically analysing a decision tree, it can be decomposed into L geometrical regions R_k over the input space. As stated in Section 2.2.1, each region R_k can be defined as $R_k = \{\alpha_{kd}^{(1)} \leq x_d \leq \alpha_{kd}^{(2)}\}_{d=1}^{M}$, where $\alpha_{kd}^{(i)} \in \bar{\mathbb{R}}$ are the boundaries[4]. As each region R_k has a single predicted label, the decision boundary is therefore the union of all $R_{k_1} \cap R_{k_2}$ with class(k_1)$\neq$ class(k_2), where class(k) is the predicted class of the region indexed by k. Therefore the unsigned distance to decision boundary of an instance $\boldsymbol{x}$ to a decision tree is

$$|d_{\boldsymbol{\theta}}(x)| = \min_{k|\boldsymbol{x}\notin R_k,\text{class}(k)\neq\hat{y}} ||\boldsymbol{x} - \text{Proj}_{R_k}(\boldsymbol{x})||_2, \tag{4.2}$$

where $\hat{y}$ is the predicted class of $\boldsymbol{x}$ by a given tree, $\text{Proj}_{R_k}(\boldsymbol{x})$ is the projection of $\boldsymbol{x}$ onto the region R_k.

We can rewrite the distance to the region R_k as

$$||\boldsymbol{x} - \text{Proj}_{R_k}(\boldsymbol{x})||_2^2 = \min_{\boldsymbol{y}} ||\boldsymbol{x} - \boldsymbol{y}||_2^2 \text{ s.t. } \boldsymbol{y} \in R_k \tag{4.3}$$

$$= \min_{y_d, d=1..M} \sum_{d=1}^{M} (x_d - y_d)^2 \text{ s.t. } \alpha_{kd}^{(1)} \leq y_d \leq \alpha_{kd}^{(2)}, d = 1..M. \tag{4.4}$$

[4]Note that the upper boundary $\alpha_{kd}^{(2)}$ can be $+\infty$ and the lower boundary $\alpha_{kd}^{(1)}$ can be $-\infty$, whenever relevant.

We can see from Eq. 4.4 an optimisation problem with a separable objective function and separable constraints. Therefore, it is possible to decompose the problem in M independent optimisation problems

$$\min_{y_d}(x_d - y_d)^2 \text{ s.t. } \alpha_{kd}^{(1)} \leq y_d \leq \alpha_{kd}^{(2)},$$

which have the solutions $y_d = \begin{cases} 0 \text{ if } \alpha_{kd}^{(1)} \leq x_d \leq \alpha_{kd}^{(2)}, \\ \alpha_{kd}^{(1)} \text{ otherwise.} \end{cases}$

One can see that, it is possible therefore to compute a projection onto a region (defined by each leaf) by traversing the tree along the corresponding decision path and set $y_d = \alpha_{kd}^{(1)}$ only if the instance $\boldsymbol{x}$ does not satisfy the splitting rule. The same analogy has been used by Alvarez *et al.* (2007) to estimate distance-based probability estimates for decision trees.

4.4 Boundary-based Fairness Constraints on Decision Trees

From now, given a decision tree, we know how to estimate the distance to the decision boundary of instances. It is therefore possible to compute the unfairness proxy measures of Eq. 4.1. Unlike the global optimisation of Eq. 4.1, we propose to learn decision trees with a top-down greedy approach using the following boundary-based fairness-aware heuristic for the disparate impact (DI) unfairness

$$H_{DI}(j,t) = \frac{\#\mathcal{D}^L(j,t)}{\#\mathcal{D}} c\left(\mathcal{D}^L(j,t)\right) + \frac{\#\mathcal{D}^R(j,t)}{\#\mathcal{D}} c\left(\mathcal{D}^R(j,t)\right) + \lambda \Big|\text{Corr}\big(z, d_{\boldsymbol{\theta}}(.)\big)\Big| \tag{4.5}$$

where (see Section 2.2.1), $\mathcal{D}^L(j,t)$ (resp. $\mathcal{D}^R(j,t)$) is the subset of instances from the parent node that satisfy the condition $X_j \leq t$ (resp. $X_j > t$), $c(.)$ is an impurity function, λ is a hyper-parameter that permits the control over the unfairness, and $\boldsymbol{\theta}$ represent all decomposed regions R_k of the currently possible tree (considering the possible update with the feature j and threshold t at the current node). The corresponding version for the boundary-based unfairness-aware heuristic for the disparate mistreatment (w.r.t. overall misclassification) is

$$H_{DM}(j,t) = \frac{|\mathcal{D}^L(j,t)|}{|\mathcal{D}|} c\left(\mathcal{D}^L(j,t)\right) \tag{4.6}$$

$$+ \frac{|\mathcal{D}^R(j,t)|}{|\mathcal{D}|} c\left(\mathcal{D}^R(j,t)\right) + \lambda \Big|\text{Corr}\big(z, \min\left(0, y\, d_{\boldsymbol{\theta}}(.)\right)\big)\Big|. \tag{4.7}$$

Instead of learning using a depth-first-builder strategy as in Algorithm 2.1, we learn our trees using a breadth-first-builder strategy in order to have a better greedy estimate of $\boldsymbol{\theta}$, which is used to compute either H_{DI} or H_{DM}. Algorithm 4.1 is a high-level algorithm, which describes steps to learn decision under the boundary-based fairness constraints. From now, we will call our trees, boundary-based fairness trees (BDTs).

Regarding the optimisation done in Algorithm 2.1, i.e., finding the feature j and threshold t_j that minimises locally (at each internal node) either the loss on Eq. 4.5 or on Eq. 4.6, Line 8 of this algorithm shows that we need to simulate the tree before computing the distance of instances to decision boundary $d_{\boldsymbol{\theta}}(.)$. While simulating the tree for this couple (j, t_j) is a constant operation, computing the distance to decision boundary takes (in the worst case) $\mathcal{O}(L \times M)$ operations per instance, i.e., $\mathcal{O}(L \times M \times N)$ in total, where N is the number of instances, M is the number of features and L is the maximum number of leaves. As a result, the implemented in Python of Algorithm 2.1 is very slow for datasets where either M or N is very high. To speed up the implementation, we implemented this algorithm using Cython.

4.5 Experimental Results

This section presents and discusses the results obtained for decision trees under boundary-based fairness constraints. Before presenting these results in Section 4.5.4, Section 4.5.1 presents the experimental setup, Section 4.5.2 presents the metrics used to evaluate learned models and Section 4.5.3 presents the baseline methods.

4.5.1 Experimental Setup

This section describes the datasets, metrics used to obtain the results.

4.5.1.1 Datasets

We use a synthetic dataset and two popular real-world datasets to compare the fair learning algorithms in the context of classification.

The first dataset is an artificial dataset for binary classification that we called *synthetic*. It is built similarly to what is proposed in Zafar *et al.* (2019), i.e., by drawing 3,000 instances from two different 2-d multivariate normal distributions, and a binary feature z correlated to the class label using a Bernouilli distribution.

Algorithm 4.1 Iterative procedure to learn a classification tree under boundary-based fairness constraint

```
Input: training set $\{\mathbf{x}_i, y_i\}_{i=1}^N$, maximum number of leaves $L$
Output: A decision tree $f_{\boldsymbol{\theta}}$, with $\boldsymbol{\theta} = \{(R_k, w_k), k = 1..L\}$
 1: Initialise an empty tree structure $T_s$
 2: Create a root_node root
 3: Create a list of splittable nodes ListNodes
 4: Add root_node in ListNodes and in $T_s$
 5: while Not StoppingCriteria do
 6:     for node in ListNodes do
 7:         for $(j, t_j)$ in (List_features, List_threshold) do
 8:             Simulate the adding of node with the split $X_j \leq t$ on the tree $T_S$
 9:             Compute $\left|\text{Corr}\big(z, d_{\boldsymbol{\theta}}(\mathbf{x})\big)\right|$ given the simulated tree
10:             Evaluate $H_{DI}(j, t)$ or $H_{DM}(j, t)$
11:         end for
12:         Keep the best $(j^*, t^*))$ w.rt. $H_{DI}$ or $H_{DM}$ for node
13:     end for
14:     Get the best node* w.r.t. $H_{DI}$ or $H_{DM}$ over ListNodes
15:     Create left_node and right_node for node*
16:     Remove node* from ListNodes
17:     Add left_node and right_node in $T_s$
18: end while
```

The second dataset is the well-known *German Credit* dataset (Du and Zhan, 2002) and the goal is to predict whether a customer should be granted credit. This dataset is very used because it is well-known that younger applicants of credit are disadvantaged due to several reasons that include the lack of historic credit results (Kallus and Zhou, 2018; Yang *et al.*, 2020).

The last dataset used is the *COMPAS* (we will sometimes refer it as *Compas*) dataset (Angwin *et al.*, 2016). COMPAS was historically a proprietary model that was used by the United State criminal justice to score criminal defendants for the risk of recidivism (Yang *et al.*, 2020) and several studies have shown that COMPAS was racially biased against African American defendants (Yang *et al.*, 2020; Kallus and Zhou, 2018; Jeff *et al.*, 2016). For this dataset, similarly as Yang *et al.* (2020), we consider the binary classification task of predicting whether a defendant profiled by COMPAS is at high risk.

Table 4.1 summarises the three datasets mentioned above, along with the associated size, dimension, sensitive feature and classification task.

Name	N	M	Sensitive feature	Task
Synthetic	3,000	3	$z = 0$ / $z = 1$	$y = -1$ / $y = 1$
German Credit	1,000	20	Age ≥ 25	Good / Bad Credit
COMPAS	$6,172$	10	African American or not	High or Not high risk

Table 4.1: Details of the datasets used for our experiments.

4.5.2 Metrics

We evaluate the performance of decision trees w.r.t. accuracy (percentage of well-classified instances) and unfairness measures. Depending on the constraint, the two unfairness measures that are popular are the disparate impact (DI) and disparate mistreatment (DM) (w.r.t. overall misclassification). W assess these measures through differences as following. If $\hat{Y}$ is the predictor, Z is the sensitive feature, Y is the true label, then $DI = \left|p\left(\hat{Y} = 1|Z = 0\right) - p\left(\hat{Y} = 1|Z = 1\right)\right|$ and $DM = \left|p\left(\hat{Y} \neq Y|Z = 0\right) - p\left(\hat{Y} \neq Y|Z = 1\right)\right|$. They quantify the degree of dependence between the sensible feature and predictions of the models. These unfairness measures are estimated using training/testing sets.

4.5.3 Methods and Hyperparameters

Our boundary-based unfairness trees (BDTs) are compared with two baselines. The basic method (without fairness constraint) is CART. It is the equivalent of setting the hyper-parameter $\lambda = 0$ on our method. The second baseline is the recent method called FAHT (Zhang and Ntoutsi, 2019). We have chosen this baseline as it is the only one we found with an open-source implementation, which allows for reuse. We re-implemented FAHT in Python because the original implementation is in Java and it is only for the disparate impact unfairness. However, thanks to its simplicity, FAHT only changes the heuristic (to be maximised) called *fair information gain* (FIG) using the formula $FIG = \begin{cases} \text{InfoGain if unfairnessGain} = 0, \\ \text{InfoGain} \times \text{unfairnessGain otherwise.} \end{cases}$

CART and FAHT only have one hyperparameter, the maximum number of leaves. BDTs also have λ as hyperparameters. Selecting best hyper-parameters that balance well two (or more) measures is not a straightforward task. To be fair with all methods, we use the technique of Conti *et al.* (2009), which consists in choosing hyperparameters

$$\beta = \arg\min_{\beta} \max\{\text{AccuracyRank}(\beta), \text{ConstraintRank}(\beta)\}. \tag{4.8}$$

This strategy only aims to guarantee that there is no better other hyperparameter on the two measures on the validation sets. We also set possible values of hyperparameters using GridSearch.

We use the $80 - 20\%$ split for training and testing sets and we computed performances on 10 independent runs.

4.5.4 Results and Discussion

This section tries to answer two major questions. (**Q1**) How do BFTs perform w.r.t. CART and FAHT on disparate impact, accuracy and disparate mistreatment? (**Q2**) Is there exist any (experimental) trade-off between unfairness and accuracy, and how easy is to find this trade-off?

Regarding **Q1**, Table 4.2 and Table 4.3 reports accuracy and unfairness evaluations. From Table 4.2, it appears that on the *Synthetic* and the *German credit* datasets, BDTs are very similar to CART w.r.t. disparate impact. This means that the strategy in Eq. 4.8 has chosen an hyperparameter (we will also argue why later) that has favoured the accuracy over the unfairness. However, on the Compas dataset, interestingly, DBTs have a lower unfairness than CART while

Datasets	Methods	DI_train	DI_test	acc_train	acc_test
Synthetic	BDT	44.596 (1.423)	40.019 (3.781)	89.509 (0.523)	87.407 (1.173)
	FAHT	0.996 (0.556)	1.951 (1.208)	61.657 (0.783)	60.889 (2.852)
	CART	40.62 (2.576)	42.84 (5.179)	88.042 (0.833)	87.5 (1.061)
German credit	BDT	7.672 (1.801)	11.401 (4.937)	80.403 (0.487)	70.611 (2.395)
	FAHT	0.0 (0.0)	0.0 (0.0)	70.472 (0.548)	68.111 (2.191)
	CART	7.044 (6.194)	5.669 (5.221)	74.528 (1.015)	70.611 (3.48)
Compas	BDT	1.29 (1.138)	2.207 (0.937)	82.135 (0.271)	81.808 (0.847)
	FAHT	0.167 (0.073)	0.358 (0.259)	81.525 (0.285)	81.889 (0.873)
	CART	7.119 (2.132)	6.561 (2.169)	83.521 (0.179)	82.852 (0.813)

Table 4.2: Accuracy and disparate impact evaluation (mean and standard deviation in %). DI_train and DI_test refers respectively to the disparate impact on training and testing sets whereas acc_train and acc_test refer respectively to the training and testing accuracy.

Datasets	Methods	DM_train	DM_test	acc_train	acc_test
Synthetic	BDT	1.959 (0.842)	2.317 (1.516)	88.546 (0.411)	86.667 (1.217)
	FAHT	36.906 (0.93)	38.117 (2.667)	55.68 (0.574)	55.34 (1.27)
	CART	2.691 (1.033)	2.747 (1.783)	88.042 (0.833)	87.5 (1.061)
German credit	BDT	7.362 (2.495)	8.516 (7.252)	80.325 (1.036)	73.05 (1.606)
	FAHT	15.392 (1.67)	13.135 (6.691)	70.25 (1.012)	69.0 (4.05)
	CART	7.871 (3.718)	12.309 (8.16)	74.528 (1.015)	70.611 (3.48)
Compas	BDT	1.959 (0.842)	2.317 (1.516)	88.546 (0.411)	86.667 (1.217)
	FAHT	16.46 (0.43)	17.358 (1.728)	81.489 (0.173)	81.368 (0.691)
	CART	2.691 (1.033)	2.747 (1.783)	88.042 (0.833)	87.5 (1.061)

Table 4.3: Accuracy and disparate mistreatment evaluation (mean and standard deviation in %). DM_train and DM_test refer respectively to the disparate mistreatment on training and testing sets whereas acc_train and acc_test refer respectively to the training and testing accuracy.

having the same level of accuracy as CART. In all cases, compared to BDT, FAHT is very effective in decreasing the disparate impact but also can seriously harm accuracy in case of severe bias like for the *Synthetic* dataset and to some extent the Compas dataset.

Still regarding **Q1**, Table 4.3, for the three datasets, BDT (slightly) reduces the unfairness w.r.t. disparate mistreatment while keeping the same level of accuracy than CART. On all the three datasets, Table 4.3 shows that FAHT does not work well with the disparate mistreatment. We found that this is partially due to the fact that disparate mistreatment is difficult to enforce on the top nodes of tree, therefore, there are usually no split rules that improve the gain. Indeed, an overfitted tree has zero disparate mistreatment on the training set.

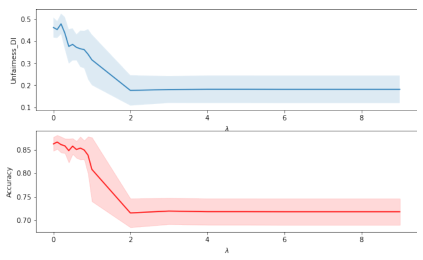
Regarding **Q2**, Figure 4.1 and Figure 4.2 show the sensitivity of accuracy and unfairness w.r.t. the hyperparamter λ whose intended goal is to balance the two objectives. From Figure 4.1 i.e., in the case of disparate impact, it appears that λ plays entirely its role since when increasing λ, the unfairness usually decreases on all the three datasets. This explains why for **Q1** and on Table 4.2, we said that the disparate impact was somehow important because the hyperparameter selection favoured accuracy. Therefore, in practical situation, λ can be therefore leveraged to balance the two objectives in the case of disparate impact unfairness.

For disparate mistreatment, still regarding **Q2**, Figure 4.2 shows that disparate mistreatment usually decreases w.r.t. λ only for *Synthetic* and *Compas* datasets, especially in the interval $[0, 1]$. For German credit the curve is very sensitive w.r.t. λ without a clear trend in $[0, 1]$. Indeed, this may be explained by the size of the dataset, which is smallest (see Table 4.1) and the empirical correlation is not a strong estimation for the *true* correlation.

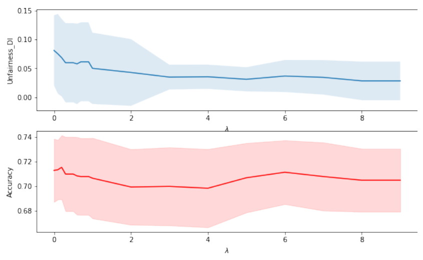
Only partially related to **Q2**, from Figure 4.1 and Figure 4.2, it appears that the accuracy does not always monotonically decreases when λ increases. This means that sometimes the accuracy is not in conflict with fairness. This has also been theoretically investigated by Wick *et al.* (2019), who precisely make some reasonable assumptions under which fairness and accuracy are not in tension. Furthermore, in practice, similar behaviours have been observed for complex models such as neural networks (Delobelle *et al.*, 2021).

4.6 Conclusion

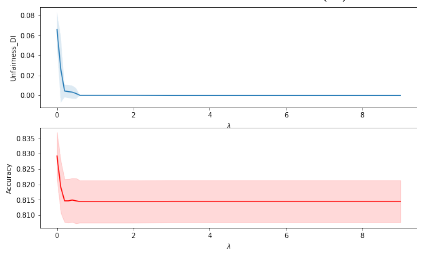
This chapter focuses on a boundary-based fairness constraint on top-down greedy decision tree learners. The description presented here was based on the published paper entitled “Boundary-Based Fairness Constraints in Decision Trees and Random Forests” (**Nanfack** *et al.*, 2021a). The main idea was the adaption of the work of Zafar *et al.* (2019), which only targeted differentiable models that have explicit expression of the distance to the decision boundary. Through a geometrical view of decision trees that are unfortunately not differentiable, we made it possible to learn decision trees under boundary-based fairness constraints. The results show that the hyperparameter λ can usually (especially for disparate impact unfairness) be leveraged to balance accuracy and fairness measures. We also compare with a recently published state-of-the-art and fair decision tree learner (FAHT) and found that our method has the advantage of being more flexible by working better



(a) Synthetic.



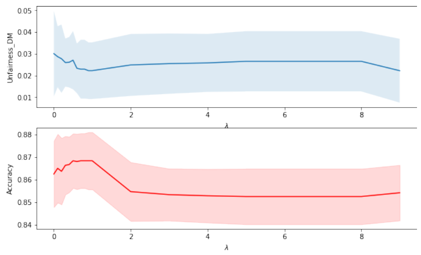
(b) German credit.



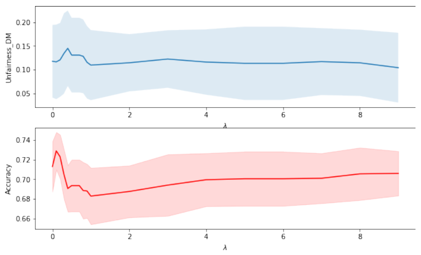
(c) Compas.

Figure 4.1: Sensitivity of disparate impact and accuracy w.r.t. the hyperparamter λ. The blue (resp. red) curve shows average of testing disparate impact unfairness (resp. accuracy) over 10 independent runs in function of the hyperparamter λ on the $x-$axis. The maximum number of leaves has been set to 7.

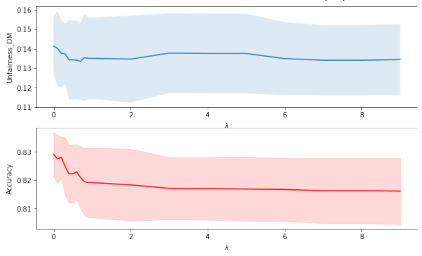
for both fairness measures. However, FAHT remains a very effective method to drastically reduce the disparate impact unfairness. Although the description and the experiments provided in this chapter focus on decision trees, the extension on random-forests is straightforward. In any case, the approach has two limitations that can be improved. First, since we are optimising a *local* heuristic which has a *global* hyperparameter, this hyperparameter is susceptible to not always play its intended role. A solution for this limitation may be to adapt locally the hyperparameter (e.g., using the number of instances involved at each splitting node). Second, unlike standard heuristics that are valid impurity functions, which have guarantees under reasonable assumptions (e.g., weak learnabiltiy i.e., the split rule performs better than random guess)(Kearns and Mansour, 1999), our approach lacks a theoretical guarantee w.r.t. accuracy and unfairness. Future studies should therefore be done to provide this guarantee.



(a) Synthetic.



(b) German credit.



(c) Compas.

Figure 4.2: Sensitivity of disparate mistreatment and accuracy w.r.t. the hyperparameter λ. The blue (resp. red) curve shows average of testing disparate mistreatment unfairness (resp. accuracy) over 10 independent runs in function of the hyperparameter λ on the $x-$axis. The maximum number of leaves has been set to 7.

Chapter 5

Constraint Enforcement on Decision Trees Using Linear Programming

This chapter presents a framework to learn decision trees under a broad class of domain-knowledge constraints that also include the fairness constraints. A part of this chapter is dedicated to formalising constraints before being able to enforce them on our decision trees. The chapter is largely based on the paper entitled "Learning Customised Decision Trees for Domain-knowledge Constraints" (**Nanfack** *et al.*, 2022b), which is currently *under revision* in the Pattern Recognition journal. In this chapter, unless otherwise stated, all figures and tables are from **Nanfack** *et al.* (2022b).

Contents

5.1 Description of the Problem, Motivation and Related Work

This section begins by motivating the work done in this chapter and by presenting the problem tackled in Section 5.1.1 before describing the most related techniques in Section 5.1.2.

5.1.1 Motivation and Problem

In several critical domains, machine learning models such as decision trees do not only need to provide the best performance, but they may also be expected to meet ethical (e.g., fairness studied in Chapter 4) and domain-specific requirements (Floridi, 2019). For example, to predict heart disease, similarly to a medical doctor, learned decision trees should take into consideration domain constraints like economical criteria when selecting features. Indeed, when medical doctors receive patients who suffers from a disease, they do not immediately ask patients to perform a very costly test such as the scanner test. Instead, they ask patients simple questions and ask them to perform simple tests to discriminate the simple cases. Only if they are not able to provide a diagnosis after this step, they recommend expensive tests (Núñez, 1991). As highlighted by Kononenko *et al.* (1997), if decision trees fail to embed such domain knowledge, medical doctors may not understand the logic of the models and thus are not likely to trust them (Kononenko *et al.*, 1997; López-Vallverdú *et al.*, 2007). On the other hand, the learned trees are also expected to provide new insights that medical doctors have not seen before in an explicit form (Kononenko *et al.*, 1997).

To learn more trustworthy models, constraint enforcement represents a theoretically supported framework to customise the shape of hypothesis set of models under a wide range of settings. However, designing learning algorithms may be complex, even for decision trees, when ethical and domain constraints must be met. As discussed in Chapter 3 (Section 3.4), recently, several decision tree methods (we will zoom on these works in Section 5.1.2)(Verwer and Zhang, 2019; Bertsimas and Dunn, 2017; Narodytska *et al.*, 2018; Aghaei *et al.*, 2019) have proposed to formalise the optimal decision tree problem as a mixed integer programming (MIP), a satisfiability (SAT) or a dynamic programming problem whose solutions often lead to small and accurate trees. A strong emphasis has been put on the speed of optimisation, neglecting the possibility to easily model prior and domain knowledge. Hence, these current methods are little to no applicable in settings where prior knowledge needs to be formalised, enforced in order to learn more comprehensible and trustworthy trees. Yet, in domains as critical as medicine and justice, the satisfaction of domain-knowledge constraints may be as important as providing a good level of accuracy (Dziugaite *et al.*, 2020; Ribeiro *et al.*, 2016; Barredo Arrieta *et al.*, 2020; Kononenko *et al.*, 1997). This suggests that a broader family of constraints should be considered when modelling the decision tree learning problem.

In this chapter, we therefore model the optimal decision tree problem so as to easily formalise constraints while allowing to explicitly control their complexity through both the number of leaf nodes and the maximum depth. We present a tree representation, based on constrained matrices, that leads to two generic linear formulations of the optimal decision tree problem. The first formulation targets binary features while the second one copes with continuous features without the need for discretisation. Moreover, in contrast to recent depth-centric models, our tree representation is a branch-like model. We show its flexibility to straightforwardly formalise and enforce a broad class of constraints that include but are not limited to ordering on features, exclusion of features over branches, feature costs and fairness.

5.1.2 Overview on Closely Related Work

Building upon standard top-down induction trees such as CART (Breiman *et al.*, 1984) and C4.5 (Quinlan, 1993), many works have attempted to enforce constraints on decision trees by learning in a greedy top-down fashion. One classical approach is to learn a possibly unconstrained decision tree and then prune it such that it satisfies the constraint. However, as discussed in Section 3.5.1, (post) pruning methods usually fail to discover richer trees since the top nodes of pruned trees do not usually change. Additionally, these trees may sometimes appear unnatural and it is possible to find better trees even manually (Piltaver *et al.*, 2016). Readers are referred to Secition 3.4.1 for a more in-depth analysis over the ability of the general top-down greedy approach to enforce constraints on decision trees.

Instead of using the greedy approach, several works have proposed to learn optimal solutions under tree-structure constraints by for example enumerating possible solutions via dynamic programming. The following paragraphs analyse these works.

Nijssen and Fromont (2007) present the DL8 algorithm using dynamic programming. Later, it is transposed into a more general framework (Nijssen and Fromont, 2010) which uses an item-set mining approach. They are able to learn optimal decision trees for different types of constraints (e.g., size of the tree and test costs). As stated by the authors, DL8 needs memory to encapsulate the huge amount of item-sets. To address the limitations of DL8, Aglin *et al.* (2020) propose DL8.5 that focuses on the task of minimising the misclassification error under depth constraint. To enforce the depth constraint, the authors had to drastically modify the DL8 algorithm. Authors highlight that "*optimisation [of DL8] is hard to combine with a constraint on the depth of a tree*" (Aglin *et al.*, 2020), to argue why they had to completely change DL8. As a result, the size of the tree is left aside and cannot be constrained, at the profit of accelerating learning through a branch-and-bound search. Moreover, without requiring another drastic change, DL8.5 cannot enforce constraints over test costs or the hierarchy of features which are both tree-structure dependent constraints (Nijssen and Fromont, 2010).

Inspired by Angelino *et al.* (2018), Hu *et al.* (2019) proposed optimal sparse

Constraints/Setting	BinOCT	Verhaeghe *et al.* (2019)	DL8.5	OSDT[1]
Number of leaves	✗	✗	✗	✓
Depth	✓	✓	✓	✓
Hierarchy/ordering	✓	✓	✗	✓
Features/Misclassification costs	✗	✗	✗	✗
Minimum #instances in the same leaf	✗	✓	✓	✗
Instances belonging to the same leaf	✓	✓	✗	✗
Range of threshold splits cont. features	✗	✗	✗	✗
Multiclass setting	✓	✗	✓	✗

Table 5.1: Comparison of state-of-the-art tree learning methods in terms of their ability to enforce the constraints considered in this paper. ✓ (resp. ✗) indicates that the current method allows (resp. does not allow) the integration of a specific constraint or is (resp. is not) able to deal with the multiclass setting.

decision trees (OSDT) that also use branch-and-bound search, but are limited to binary classification. Analytic bounds are used to prune the search space while the number of leaves is constrained using a regularised loss function that balances accuracy and the number of leaves. Because OSDT and its extended version GOSDT (Lin *et al.*, 2020) use a customised branch-and-bound search, similarly to DL8.5, integrating tree-structure dependent constraints that are not directly expressed in the objective function will need to completely accommodate the learning algorithm. Nonetheless, our work shares similarities with OSDT since our model also allows constraining directly the number of leaves.

Because of the improved computational speed of machines, several works (Bertsimas and Dunn, 2017; Narodytska *et al.*, 2018; Verwer and Zhang, 2019; Aghaei *et al.*, 2019; Avellaneda, 2020) propose to exploit MIP and SAT solvers to learn optimal decision trees by constraining the depth. More specifically, Narodytska *et al.* (2018) and Avellaneda (2020) use SAT solvers to learn the smallest tree for perfect classification. We do not tackle the same problem in this chapter. Bertsimas and Dunn (2017) present a MIP formulation of the optimal decision tree problem for a given depth. Their model (called OCT) handles both univariate and multivariate splits. As stated in Hu *et al.* (2019), OCT is not easily reproducible and no public code is available (Hu *et al.*, 2019).

[1]GOSDT (Lin *et al.*, 2020), an improved version of OSDT, does multiclass but only through

A more recent MIP formulation (BinOCT) has been proposed by Verwer and Zhang (2019) to be efficient in computations. To speed up optimisation, BinOCT considers full and complete binary trees under depth constraints. This raises a question about the optimality of their learned trees when the optimal solution is not full and complete. In particular, in presence of domain constraints, BinOCT will not find a solution if trees do not need to be full and complete. Moreover, in order to accelerate optimisation, BinOCT does not keep track of misclassified instances through decision variables, making it impractical to change objective function in order to integrate instance-dependent constraints like misclassifition costs. The work of Verhaeghe *et al.* (2019) uses constraint programming to learn optimal decision trees, but only under depth constraint, and targets only binary classification.

To date, previous models have focused on accelerating the learning of *optimal* decision trees. Our work instead focuses on the enforcement of a broad class of constraints for trustworthiness. As it is not the focus of previous works, they do not natively offer mechanisms to do that. For example, they may allow to set a depth constraint, but still, these trees may not meet domain-knowledge constraints. Such decision trees may be rejected by domain experts (Freitas, 2014; Barredo Arrieta *et al.*, 2020). Table 5.1 shows a representative sample of types of constraints that the above methods are (or are not) able to handle if domain constraints need to be imposed. For example, BinOCT, DL8.5 and Verhaeghe *et al.* (2019) are depth-centric models that are not directly formulated to find the optimal decision tree for a fixed number of leaf nodes. Also, while BinOCT with its discretisation heuristic can cope with continuous features, all other methods require binary features. As a result, they (BinOCT included) cannot integrate the constraint that threshold splits must belong to a given interval, which is useful when seeking for relevant decision rules with respect to the domain expertise (Liu *et al.*, 2002; Verbakel *et al.*, 2015; Shaharanee *et al.*, 2011). Some of the above methods could be modified at a significant cost to take into account some of the considered constraints (or even other ones). However, as they are not designed for that purpose, it is neither formally nor technically obvious how to do so (e.g., transition from DL8 to DL8.5). This motivates our work to make constraint enforcement easier, thanks to a specifically designed tree representation.

The above analysis of the state of the art shows the need for a general framework to: i) efficiently handle complexity control for generalisation of decision trees; ii) and learn decision trees under domain constraints that are important for safety and trustworthiness. The contributions of this chapter can be summed up as

one-vs-all.

follows: (1) we propose a tree representation based on constrained matrices that leads to two new generic linear programming formulations of the optimal decision tree problem allowing to easily integrate domain-knowledge constraints; (2) both formulations give the possibility for users to easily control complexity through the constraints on the number of leaf nodes and the maximum depth of the decision tree; (3) we theoretically show that elements of these matrices give the ability to easily formalise domain knowledge as constraints to improve trustworthiness.

The rest of the paper is organised as follows: Section 5.2 introduces our tree representation and our first generic linear programming formulation for binary features, Section 5.3 presents the second formulation, which handles continuous features without discretisation; Section 5.4 formalises the expression of domain-knowledge constraints; Section 5.5 shows use cases for real-word applications; Section 5.6 benchmarks constraint-free performance of our models w.r.t. recent models and discusses the results; Section 5.8 analyses computational time before concluding.

5.2 Our Tree Representation and First Formulation to Enforce Constraints on Decision Trees

This section presents our tree representation and the first formulation that models the classification tree learning problem via binary variables and linear constraints.

Let us consider a sample where all features have been transformed into binary features (numeric features can discretised into categorical ones and, then, into binary ones). A decision tree from this dataset is characterised by its number of leaf nodes L, its maximum depth K and its size. In what follows, $\boldsymbol{X} \in \{0,1\}^{N \times M \times V}$ denotes a sample (without labels), N is the number of instances, M is the number of features of the dataset and V is the number of values which can be taken by a feature. Here, it is assumed that data only have binary features, thus $V = 2$. However, the following can be easily extended to $V > 2$ for V-array trees. $\boldsymbol{Y} \in \{0,1\}^{N \times C}$ is the matrix providing the labels of instances (into their one-hot-encoding form) and C is the number of classes. Finally, $[n]$ denotes the set $\{0, 1, ..., n-1\}$, for $n \in \mathbb{N}$. Elements of matrices will be written without bold (e.g., elements of $\boldsymbol{X}$ will be written X_{ijk} or x_{ijk}, where i, j, k are indexes).

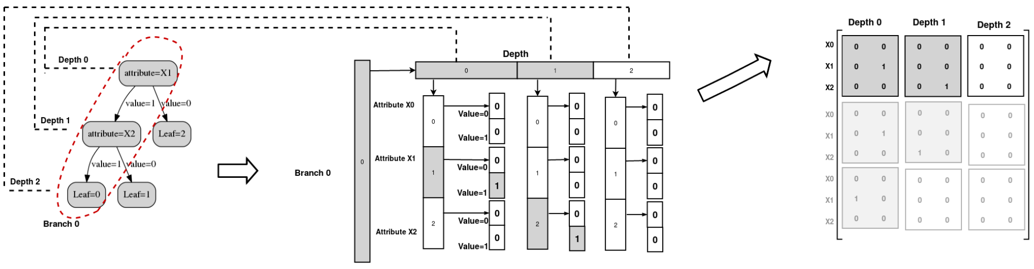


Figure 5.1: Tree encoding detailed for a branch. Left: an example of a decision tree on which depth are depicted and the first branch of the tree is circled. Center: the corresponding 4-dimensional matrix with pointers representing how to pass from a dimension to an other (only the branch 0 is highlighted here). Right: the corresponding encoding matrix $\boldsymbol{\theta}$ representation.

5.2.1 Tree Representation

We need a convenient and efficient way to encode trees so as to easily formalise domain constraints such as those related to path of decisions (e.g., ordering on tests and test costs on branches). Rather than a tree representation that decomposes a tree into a set of nodes, our decision trees are decomposed in terms of set of branches. We propose to encode branches with the (multidimensional) matrix [2] $\boldsymbol{\theta} \in \{0,1\}^{L\times K\times M\times V}$. To illustrate, $\theta_{ijlp} = 1$ if the p-th value of the l-th feature is selected, at depth j of the i-th branch. Therefore, $\boldsymbol{\theta}_{i,*}$ encodes the i-th branch of the tree, $\boldsymbol{\theta}_{ij,*}$ encodes the selection of features and its values at depth j of branch i, and $\boldsymbol{\theta}_{ijl,*}$ is the vector encoding whether the l-th feature has been selected[3] at depth j of branch i. Figure 5.1 shows an example of a decision tree with its encoding matrix.

From this description, $\boldsymbol{\theta}$ can be represented as a 4-dimensional matrix. The following constraints need to be added in order to encode the selection of features over branches of trees.

- The same feature is chosen at the top of all branches:

$$\forall i \in [L], a \in [L], l \in [M] \colon \sum_{p\in[V]} \theta_{i,0,l,p} = \sum_{p\in[V]} \theta_{a,0,l,p}. \tag{5.1}$$

[2]For simplicity, instead of multidimensional matrices, we will use the term matrix.

[3]The l-th feature is selected if at least one its value is $\neq 0$.

- No more than one feature must be chosen on a branch i, at depth j:

$$\forall i \in [L], j \in [K]\colon \sum_{l\in[M]} \sum_{p\in[V]} \theta_{ijlp} \leq 1. \tag{5.2}$$

- Each feature is chosen at most once on a branch:

$$\forall i \in [L], l \in [M]\colon \sum_{j\in[K]} \sum_{p\in[V]} \theta_{ijlp} \leq 1. \tag{5.3}$$

- For every branch, if a feature is chosen at depth j, on all previous depths $0, ..., j-1$, a feature must be selected:

$$\forall i \in [L], j \in [K-1]\colon \sum_{l\in[M]} \sum_{p\in[V]} \theta_{ijlp} \geq \sum_{l\in[M]} \sum_{p\in[V]} \theta_{i(j+1)lp}. \tag{5.4}$$

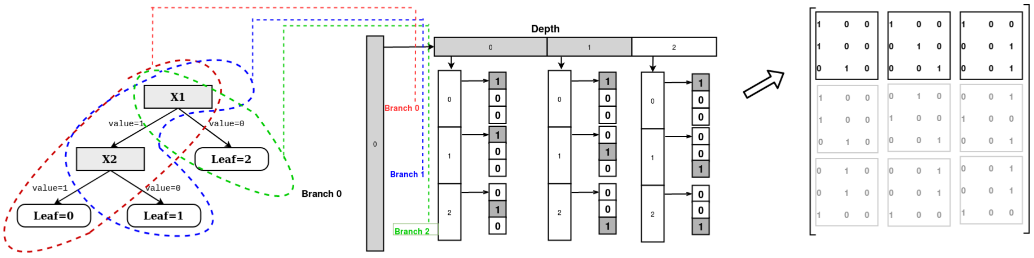


Figure 5.2: Relationship between branches (matrix $\boldsymbol{\alpha}$). Left: branches of the tree. Center: relationship between the first branch (branch 0) and the other branches; the deepest vector (with 3 bits horizontally aligned) defines the 3 possibilities of pair of branches given a depth: (1) branches are equal up to that depth, (2) branches are siblings, (3) branches are different with a variable or a value before the given depth. Right: the corresponding matrix $\boldsymbol{\alpha}$ which encodes the relationship between all the branches.

From what has been described, $\boldsymbol{\theta}$ models a set of L branches with at most K features selected per branch. However, for $\boldsymbol{\theta}$ to represent a valid tree, relations between branches need to be defined. In particular, two branches i and a are *siblings at depth* j if all features and values that are selected on branch i and a until depth j, are equal, but on depth j, chosen features remain the same between the two branches and only their values differ (e.g., branch 1 and 2 at depth 0). Formally, branch relations are encoded using the 3-dimensional matrix $\boldsymbol{\alpha} \in \{0,1\}^{L\times K\times L\times 3}$ and we define three types of relations: (i) $\alpha_{ija,0} = 1$ if all the features and their values on branches i and a are equals up to and including depth j; (ii) $\alpha_{ija,1} = 1$ if branches i and a are siblings at depth j; and (iii) $\alpha_{ija,2} = 1$ if it exists one feature or one value of feature selected on depth $j_1 < j$ which differs from branches a and i. For example, in Figure 5.2, at depth 1, branches 0 and 1 are siblings, so $\alpha_{011,1} = 1$; and at depth 1 branches 0 and 2 fall into the third case; so $\alpha_{012,2} = 1$.

With the above definitions for the $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ matrices, we can now express the constraints that a valid tree representation should satisfy.

- All branches must be pairwise different, i.e., any couple of branches falls in the third case at depth $K-1$:

$$\forall i \in [L], a \in [L] \setminus \{i\} \colon \alpha_{i,K-1,a,0} = 0. \tag{5.5}$$

- Every pair of branches matches only one of the previous cases:

$$\forall i, a \in [L], j \in [K] \colon \sum_{q \in \{0,1,2\}} \alpha_{ijaq} = 1. \tag{5.6}$$

- Every pair of branches in case (i) at depth j must have selected on depths $j_1 \leq j$ the same feature and have the same values of those features:

$$\forall i, a \in [L], j \in [K], j_1 \in [j+1], p \in [V] \colon \alpha_{ija,0} = 1 \implies \theta_{ij_1lp} = \theta_{aj_1lp}. \tag{5.7}$$

- On a branch, if no feature is chosen at a specific depth j, no other branch can be in case (i) at depth $j-1$:

$$\forall i \in [L], a \in [L] \setminus \{i\}, j \in [K-1] \colon \alpha_{ija,0} \leq \sum_{l \in [M]} \sum_{p \in [V]} \theta_{i,j+1,lp}. \tag{5.8}$$

- Every pair of branches in case (iii), which are different on at least one depth before j must have a depth where feature/value is different. This relationship appears when earlier on depth $j_1 < j$, the branches were siblings. Therefore, the constraint can be simply written as:

$$\forall i \in [L], j \in [K], j_1 \in [j], a \in [L] \setminus \{i\} \colon \alpha_{ija,2} \geq \alpha_{ij_1a,1}. \tag{5.9}$$

- If a feature is selected on a depth of a branch, the branch must have at least one sibling:

$$\forall i \in [L], j \in [K] \colon \sum_{a \in [L] \setminus \{i\}} \alpha_{ija,1} \geq \sum_{l \in [M]} \sum_{p \in [V]} \theta_{ijlp}. \tag{5.10}$$

- Every pair of sibling branches must be equal (in terms of features and values) up to depth $j-1$:

$$\forall i, a \in [L], j \in [K] \setminus \{0\}, j_1 \in [j], p \in [V] \colon \alpha_{ija,1} = 1 \Rightarrow \theta_{ij_1lp} = \theta_{aj_1lp}. \tag{5.11}$$

- Every pair of sibling branches on depth j must have the same feature chosen on this depth:

$$\forall i, a \in [L], j \in [K], l \in [M] \colon \alpha_{ija,1} = 1 \Rightarrow \sum_{p\in[V]} \theta_{ijlp} = \sum_{p\in[V]} \theta_{ajlp}. \tag{5.12}$$

- Values of features for each pair of sibling branches at depth j must be different, if the feature is selected and equal to zero otherwise:

$$\forall i, a \in [L], l \in [M], j \in [K], p_1 \in [V] \colon \alpha_{ija,1} = 1 \Rightarrow \theta_{ijlp_1} + \theta_{ajlp_1} = \sum_{p\in[V]} \theta_{ijlp}. \tag{5.13}$$

- If no feature is selected at depth j of the branch i, no branch can be a sibling of another branch i:

$$\forall i \in [L], a \in [L] \setminus \{i\}, l \in [M], j \in [K], p \in [V] \colon \alpha_{ija,1} \leq \sum_{p\in[V], l\in[M]} \theta_{ijlp}. \tag{5.14}$$

Our tree representation is composed of matrices $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ on which constraints (5.1–5.14) are applied.

Matrices $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ respectively have $L \times K \times M$ and $L^2 \times K$ variables. The previous constraints related to these matrices were derived from the formal definition of decision trees decomposed in terms of branches.. In several cases, they are paired to ensure the logical equivalence ($\Longrightarrow$ and $\Longleftarrow$ implications) of the definition (e.g., sibling branches from Eq. 5.10 to Eq. 5.14).

To learn trees, one needs to create an objective function where instance-dependent variables need to be explicitly introduced, as shown hereafter.

5.2.2 Encoding Global Objective Functions

This section introduces variables and constraints in order to encode the global objective function. Section 5.4 will show how constraints can be easily enforced thanks to these developments.

Let $\boldsymbol{\xi} \in \{0,1\}^{N\times L}$ denote the mapping between the set of examples of the dataset and the set of leaves (e.g., $\xi_{ei} = 1$ if example e belongs to the i-th leaf/branch and 0 otherwise). Let $\boldsymbol{\nu} \in \{0,1\}^{L\times C}$ denote the mapping between the set of branches and the set of classes (e.g., $\nu_{ic} = 1$ if c is the predicted class of branch i and 0 otherwise). Finally, $\boldsymbol{\epsilon} \in \{0,1\}^{N}$ defines the $0-1$ loss over the example e. For recall, $\mathbf{Y} \in \{0,1\}^{N\times C}$ represents labels of the dataset, as defined in beginning of this section.
The following constraints describe how to encode global objective functions.

- Each example must belong to one leaf:

$$\forall e \in [N]\colon \sum_{i\in[L]} \xi_{ei} = 1. \tag{5.15}$$

- If an example belongs to a leaf node, all the features/values chosen on the corresponding branch must be the same to this example:

$$\begin{aligned}&\forall e \in [N], i\in[L], j\in[K], l\in[M], p\in[V]\colon\\ &\theta_{ijlp} = 1 \Rightarrow \xi_{ei} \le 1 - x_{elp} + \theta_{ijlp}, \qquad (5.16a)\\ &\theta_{ijlp} = 1 \Rightarrow \xi_{ei} \le 1 + x_{elp} - \theta_{ijlp}. \qquad (5.16b)\end{aligned}$$

- Each branch of leaf node must predict exactly one class:

$$\forall i \in [L]\colon \sum_{c\in[C]} \nu_{ic} = 1. \tag{5.17}$$

- The class of a leaf node (or branch) is the majority vote of examples belonging to this branch:

$$\forall i \in [L], c_1, c_2 \in [C]\colon \nu_{ic_1} = 1 \Rightarrow \sum_{e\in[N]} \xi_{ei} * y_{ec_1} \ge \sum_{e\in[N]} \xi_{ei} * y_{ec_2}. \tag{5.18}$$

- The error of the predicted class of an example is equal to one if the class of its branch is different from the true class and equal to zero, otherwise:

$$\begin{aligned}&\forall e \in [N], c\in[C], i\in[L]\colon\\ &\xi_{ei} = 1 \Rightarrow \epsilon_e \ge -\nu_{ic} + y_{ec}, \qquad (5.19a)\\ &\xi_{ei} = 1 \Rightarrow \epsilon_e \le 2 - \nu_{ic} - y_{ec}. \qquad (5.19b)\end{aligned}$$

Using the above results, the misclassification error of the tree is simply

$$\sum_{e \in [N]} \epsilon_e. \tag{5.20}$$

This objective function can be used by solvers to search for an optimal tree.

The complexity $\mathcal{C}$ in terms of number of variables of the entire constraint program of our formulation is $\mathcal{O}\Big(L \times K(L+M) + N \times L\Big)$. It can be reduced to $\mathcal{O}\Big(L(L+M+N)\Big)$ for shallow (less deep) trees when prior values of L and K are known. Indeed, since the tree encoding complexity (in terms of number of variables) is $\mathcal{O}(L \times K \times M + L^2 \times K)$ and the objective function encoding is $\mathcal{O}(N \times L)$, then, the total complexity is

$$\mathcal{C} = \mathcal{O}(L \times K \times M + L^2 \times K + N \times L) = \mathcal{O}\Big(L \times K(L+M) + N \times L\Big)$$

For shallow (less deep) trees, K is small, therefore less than a fixed maximum value. So, $\mathcal{C} = \mathcal{O}(L^2 + L \times M + L \times N)$.

The time to find the optimal solution increases proportionally with the size of the dataset and the number of leaves. Another important insight from the analysis above is the fact that, in terms of number of variables, the complexity $\mathcal{C}$ highly varies depending on which term is the biggest term between N and $K(L+M)$. In particular, N is not usually tunable (except with sub-sampling) whereas M could be because of the binarisation step. Hence, in practical implementations with binary features, M should be ideally negligible w.r.t. N (i.e., a $o(N)$), when prior values of K and L are known.

Constraints 5.7, 5.11, 5.12, 5.13, 5.16a, 5.16b, 5.18, 5.19a and 5.19b are not directly expressed in a linear form. They are presented in a form called *indicator* constraints, but modern linear programming solvers generally prefer this form. Nonetheless, they can be linearised using big-M constraints. For example, the constraint 5.7 can be written as $\theta_{ijlp} - \theta_{ajlp} \leq M * (1 - \alpha_{ija,0})$, with M being a big positive number so that when $\alpha_{ija,0} = 1$, the left part of the inequality must be null.

It is worth mentioning that, in contrast to Bertsimas and Dunn (2017); Verhaeghe *et al.* (2019); Verwer and Zhang (2019), our first model does not rely on a specific type of solver since we exploit only binary variables and linear constraints. Therefore, it has the advantage to be implementable on CP solvers as well as MIP and ILP solvers. Readers are encouraged to read a brief overview of MIP techniques in Chapter 2, especially, in Section 2.5.1.2.

5.3 Second Formulation to Handle Continuous Features

Based on the previously introduced tree representation in Section 5.2, this section presents our second formulation. It copes with continuous splits, which was not possible with the first formulation by only bringing few modifications in the matrices and their associated constraints. These modifications are explained hereafter.

To go from binary splits to numerical splits, based on the proposed tree representation and the first formulation, we only need a semantic modification related to $\boldsymbol{\theta}$ and an introduction of learnable split thresholds. Now, $\boldsymbol{\theta}_{ij,*,0}$ encodes the feature selected on branch i, at depth j while $\theta_{ijl,1}$ encodes the type of relation ($\leq$ or $>$) contained on branch i, depth j, for the l-th feature. More explicitly, $\theta_{ijl,0} = 1$ if the l-th feature is selected on branch i, at depth j and $\theta_{ijl,1} = 0$ (resp. $\theta_{ijl,1} = 1$) if the l-th feature is selected and the type of relation is $\leq$ (resp. $>$). Here again, note that $\boldsymbol{\theta} \in \{0,1\}^{L\times K\times M\times V}$.

In order to make this formulation work with learnable numeric splits, we need a matrix to encode learnable threshold split values. This matrix is $\boldsymbol{\omega}$. Therefore, ω_{ij} encodes the threshold split value on branch i, at depth j. In order to bound its range of value, $\boldsymbol{\omega} \in [0,1]^{L\times K}$ and numeric features have to be scaled using the *min-max scaler*. Hence, it implies that $\boldsymbol{X} \in [0,1]^{N\times M}$.

Firstly, we show how to rewrite the expression of constraints in Section 5.2.1 in order to deal with learnable numeric splits. The constraints (5.1–5.4) that are related to features over branches of the tree can be adapted by rewriting $\sum_{p\in[V]} \theta_{ijlp}$ as simply $\theta_{ijl,0}$. As the tree representation is preserved, the matrix $\boldsymbol{\alpha}$, which encodes relationships between branches, keeps the same semantic. Except the constraints (5.5–5.6) and (5.9) which do not change, the rest of constraints are rewritten as follows.

- (5.7) $\forall i, a \in [L], j \in [K], j_1 \in [j+1], l \in [M], p \in [V]$:

$$\alpha_{ija,0} = 1 \Rightarrow \begin{cases} \theta_{ij_1lp} = \theta_{aj_1lp} \\ \omega_{ij_1} = \omega_{aj_1} \end{cases}.$$

- (5.8) $\forall i \in [L], a \in [L] \setminus \{i\}, j \in [K-1], l \in [M]$: $\alpha_{ija,0} \leq \sum_{l\in[M]} \theta_{i,j+1,l,0}$.
- (5.10) $\forall i \in [L], j \in [K], l \in [M]$: $\sum_{a\in[L]\setminus\{i\}} \alpha_{ija,1} \geq \sum_{l\in[M]} \theta_{ijl,0}$.
- (5.11) $\forall i \in [L], j \in [K], l \in [M], p \in [V], a \in [L] \setminus \{i\}, j_1 \in [j]$:

$$\alpha_{ija,1} = 1 \Rightarrow \begin{cases} \theta_{ij_1lp} = \theta_{aj_1lp} \\ \omega_{ij_1} = \omega_{aj_1} \end{cases}.$$

- (5.12) $\forall i, a \in [L], j \in [K], l \in [M]\colon \alpha_{ija,1} = 1 \Rightarrow \theta_{ijl,0} = \theta_{ajl,0}$.
- (5.13) $\forall i, a \in [L], j \in [K], l \in [M]\colon \alpha_{ija,1} = 1 \Rightarrow \begin{cases} \theta_{ijl,1} + \theta_{ajl,1} = \theta_{ijl,0} \\ \omega_{ij} = \omega_{aj} \end{cases}$.

Secondly, once a decision tree has been encoded through matrices $\boldsymbol{\alpha}$, $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$, we need to encode objective functions in order to learn decision trees. This is done by keeping all constraints in Section 5.2.2 unchanged except constraints 5.16a and 5.16b that are rewritten as

$$\forall e \in [N], i \in [L], j \in [K], l \in [M]\colon \alpha_{ijl,0} = 1 \Rightarrow \begin{cases} \xi_{ei} \leq 1 + \omega_{ij} - x_{el}(1 - \theta_{ijl,1}) \\ \xi_{ei} \leq 2 - \omega_{ij} - \delta + (x_{el} - 1)\theta_{ijl,1}, \end{cases}$$

where $\delta \approx 0$ and is a positive number to ensure the strict inequality option "$>$" for splits. The first constraint states that a data-instance cannot belong to the leaf corresponding to branch i if its feature value x_{el} is larger than the threshold ω_{ij} at depth j. This corresponds to a "$\leq$" inequality and is materialised by $\theta_{ijl,1} = 0$; the second constraint has the similar mining but with the "$>$" inequality.

5.4 Formalising Domain Knowledge as Constraints

With the tree representation and the first formulation presented in the previous section, in the following, we show how domain knowledge can be straightforwardly formalised as constraints and integrated in our models.

5.4.1 Ordering of Features

It often appears that domain experts have a prior belief of a specific ordering of features. This prior may directly come from their background knowledge or hard material constraints which favour asking patients to perform tests before others. As an illustration, to predict liver diseases, doctors will ask patients their age before performing a *bilirubin test* since the interpretation of this test depends on the age of the patient (Hodgson *et al.*, 2018). Therefore, a comprehensible decision tree for domain expert should encode this prior knowledge. The constraint "feature X_{l_2} must not appear before X_{l_1}" is expressed as

$$\forall i \in [L], j \in [K] \setminus \{0\}, \sum_{j_1 \in [j]} \sum_{p \in [V]} \theta_{ij_1 l_2 p} \leq 1 - \sum_{p \in [V]} \theta_{ijl_1 p}.$$

5.4.2 Test Costs on Features

Experts can also be constrained by economical considerations to ask patients to do tests before taking decisions (Núñez, 1991). A good illustration of this importance is the case of application of machine learning algorithms in domains with lack of available material resources to perform medical tests. Hence, one can specify test costs on features in order to limit the total economical cost to spend before reaching a decision. With our framework, it is possible to learn trees with a maximum classification cost τ on a cost-sensitive dataset. This constraint becomes

$$\forall i \in [L]\colon \sum_{j\in[K]} \sum_{l\in[M]} \sum_{p\in[V]} \theta_{ijlp} * \mathrm{cost(l)} \leq \tau.$$

where $\mathrm{cost}(l)$ denotes the test cost of feature $\mathbf{X_l}$.

5.4.3 Expected Cost for Classification

As for the same reasons with test costs, certain applications may require to constrain the expected classification cost. The weighted test cost of a branch i is $\frac{1}{N}\sum_{e\in[N]}\sum_{l\in[M]}\xi_{ei} * \mathrm{cost}(l) * \big(\sum_{p\in[V]}\theta_{ijlp}\big)$. As this previous term is non-linear, it is possible to linearise it with variables N_{ij} representing the weighted test cost of the branch i at depth j, with the constraints:

$$\forall i \in [L], j \in [M], l \in [L], \sum_{p\in[V]} \theta_{ijlp} = 1 \Rightarrow N_{ij} = \mathrm{cost}(l) * \sum_{e\in[N]} \xi_{ei},$$

$$\text{and } \forall i \in [L], j \in [M], \sum_{l\in[M] p\in[V]} \theta_{ijlp} = 0 \Rightarrow N_{ij} = 0.$$

Therefore, the expected cost for classification is Ecost $= \frac{1}{N}\sum_{i\in[L]}\sum_{j\in[K]} N_{ij}$. Imposing a maximum expected cost for a decision tree can be stated as

$$\sum_{i\in[L]} \sum_{j\in[K]} N_{ij} \leq N * \tau.$$

5.4.4 Number of Instances on Leaf Nodes

To reduce the growth of a tree, one can also give the minimum number of instances on leaf nodes. This can be easily done by adding

$$\forall i \in [L], \sum_{e\in[N]} \xi_{ei} \geq \text{ minNumberInstances.}$$

5.4.5 Instances that Must Be in the Same Leaf

It often appears that datasets come with instance names and domain experts may have background knowledge on specific instances. Similarly to clustering (Wagstaff *et al.*, 2001b), practitioners may want to impose that certain instances satisfy the same decision rule, i.e., belong to the same leaf node. One can also specify that two instances must be in the same leaf node with the proposed framework using the constraint expressed as

$$\forall i \in [L], \xi_{e_1,i} = \xi_{e_2,i}, \text{ if } e_1 \text{ and } e_2 \text{ has to be in the same leaf node.}$$

5.4.6 Presence or Exclusion of a Feature Over the Tree or Over a Branch

It often appears that in several situations (e.g., due to noisy data), the selected features of a decision tree do not match the domain-related relevant features (López-Vallverdú *et al.*, 2012). In such situations, domain experts may not trust the learned tree. Using our formulation, it is possible to impose the selection of a feature X_l (without even knowing where it should be selected in the tree) thanks to the constraint

$$\sum_{i\in[L]} \sum_{j\in[K]} \sum_{p\in[V]} \theta_{ijlp} \geq 1.$$

Additionally, two or more features may be redundant in a decision rule, i.e., over a specific branch. For some specific reasons, they should not be selected simultaneously in a branch (i.e., decision-wise) or in the whole tree (i.e., model-wise). Domain experts can provide this knowledge which can be infused as constraints. It can be guaranteed that two features X_{l_1} and X_{l_2} do not appear on the same branch of a decision tree with

$$\forall i \in [L], \sum_{j\in[K]} \sum_{p\in[V]} \theta_{ij,l_1,p} + \theta_{ij,l_2,p} \leq 1.$$

X_{l_1} and X_{l_2} will not appear simultaneously into the tree if

$$\sum_{i\in[L]} \sum_{j\in[K]} \sum_{p\in[V]} Q_{ij,l_1,p} + Q_{ij,l_2,p} \leq 3.$$

5.4.7 Fairness through Demographic Parity and Minimum Accuracy for a Group

5.4.7.1 Disparate Impact or Demographic Parity

In Chapter 4, we imposed fairness constraints through a top-down greedy method. The formulations presented in this chapter also allow to formalise the fairness constraint. As a reminder, demographic parity or disparate impact fairness aims to ensure that the predictions of a classifier do not depend on a sensitive feature z such as gender or race, i.e., $p(\hat{y}=1|\mathbf{x}_z) = p(\hat{y}=1|\mathbf{x}_{\bar{z}})$ (Lohaus *et al.*, 2020). In practice, the demographic parity fairness is evaluated on a sample, through the difference of demography parity (DDP) given by

$$\text{DDP} = \Big| \frac{\#\{\mathbf{x} \in \mathcal{D}; \hat{y}=1, \mathbf{x}_z = 1\}}{\#\{\mathbf{x} \in \mathcal{D}; \mathbf{x}_z = 1\}} - \frac{\#\{\mathbf{x} \in \mathcal{D}; \hat{y}=1, \mathbf{x}_z = 0\}}{\#\{\mathbf{x} \in \mathcal{D}; \mathbf{x}_z = 0\}} \Big|.$$

Let N_z (resp. $N_{\bar{z}}$) be the number of samples belonging to the first (second) category of the binary sensitive feature, N_z^+ (resp. $N_{\bar{z}}^+$) be the number of positively predicted samples belonging to the first (resp. second) category of the sensitive feature. Therefore, the DDP can be expressed as DDP $= \big|N_z^+/N_z - N_{\bar{z}}^+/N_{\bar{z}}|$. Looking back at our tree formulation, thanks to the partioning of the branch representation, we can partition samples according to branches/leaves $i \in [L]$, create variables N_{iz}^+ (resp. $N_{i\bar{z}}^+$) for each branch and express the DDP as $\big|\frac{N_{\bar{z}} * \sum_{i\in[L]} N_{iz}^+ - N_z * \sum_{i\in[L]} N_{i\bar{z}}^+}{N_{\bar{z}} * N_z}\big|$. Hence, using our tree formulation, a decision tree satisfies the τ-DDP if

$$-\tau * N_{\bar{z}} * N_z \leq N_{\bar{z}} * \sum_{i\in[L]} N_{iz}^+ - N_z * \sum_{i\in[L]} N_{i\bar{z}}^+ \leq \tau * N_{\bar{z}} * N_z,$$

where N_{iz}^+ and $N_{i\bar{z}}^+$ are variables defined thanks to the following constraints:

$$\forall i \in [L], \nu_{i,1} = 0 \Rightarrow \begin{cases} N_{i\bar{z}}^+ = 0 \\ N_{iz}^+ = 0 \end{cases}, \nu_{i,1} = 1 \Rightarrow \begin{cases} N_{i\bar{z}}^+ = \sum_{e\in[N]} \xi_{ei} * (1 - x_{ez}) \\ N_{iz}^+ = \sum_{e\in[N]} \xi_{ei} * x_{ez} \end{cases}.$$

5.4.7.2 Minimum Accuracy on an Underrepresented Group

In the same direction, if one would like to guarantee that a certain percentage τ of instances from a targeted group G should not be misclassified, it can be easily imposed using the following constraint: $\sum_{e\in G} \epsilon_e \leq (1-\tau) * |G|$.

More broadly, it is important to note that, by drawing inspiration from demographic parity measures and minimum accuracy, other constraints called in Cotter *et al.* (2019) as *rate constraints* (evaluated using input/outputs of models) can be formalised and expressed within the proposed formulation.

5.5 Use Cases with Constraint Enforcement on Decision Trees

This section demonstrates empirically the capability of our model to enforce various types of domain-knowledge constraints. We now refer to our decision trees as *CPTrees.* Following the methodology in Yang *et al.* (2020) and Cotter *et al.* (2019), we experimentally validate the ability of our framework to enforce a broad class of domain-knowledge constraints on several real-world applications with prior domain-knowledge that should be enforced. In the following, for each application, we briefly discuss the results obtained with CPTree and constrained CPTree in terms of trees and decision rules. In order to have a point of comparison, we also include CART in our experiments, although any other method benchmarked in Section 5.6 could be chosen. We report the accuracy in our experiments to study whether constraint enforcement impacts it: ideally, constrained decision trees should remain reliable and accurate. Finally, datasets are split using the $67-33\%$ train-test percentage (except on COMPAS with the $40-60\%$). We mostly use 6 as the (maximum) number of leaves in order to keep the interpretability of learned trees and we also evaluate on single run to visualise trees.

5.5.1 Ordering Constraint Applied to the Prediction of Breast Cancer Survival

The first application is related to the *Haberman's survival* dataset (Dua and Graff, 2017), which contains features like *age* and *positive_nodes* (number of positive axiliary nodes detected), etc. The goal is to predict whether a patient survives after a surgery for breast cancer. In this domain, breast cancer is extremely violent for younger patients because it can weaken the patient and make the surgery dangerous. On the other hand, older patients usually have difficulties to recover from surgeries (in general) since it puts the body under high stress. In between these two categories, patients have more chances to survive a breast cancer surgery (Tina Binesh and Sydney, 2018). Moreover, it is established that knowing the age, the evolution of *positive_nodes* is piece-wise linear (Lohaus *et al.*, 2020) and that the feature *age* precedes the feature *positive_nodes* on a causal directed graph (Li *et al.*, 2016). Therefore, a medical valid decision tree should first select the feature *age* before the feature *positive_nodes*. In other words, for this use case on

	Haberman's survival		**Diabetes**	
	Train	**Test**	**Train**	**Test**
CART	77.07	71.28	79.96	68.50
CPTree	80.97	68.31	81.90	68.89
C-CPTree	78.53	73.26	79.57	74.40

Table 5.2: Accuracy of CART, (unconstrained) CPTree and (constrained CPTree) C-CPTree.

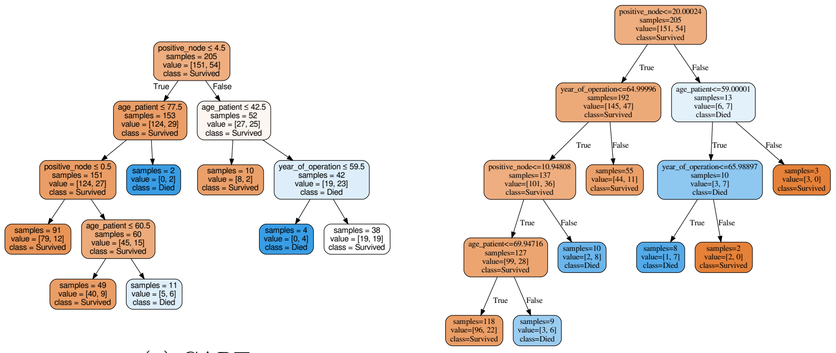
the prediction of survival after breast cancer, we impose the constraint that the feature *age* should appear before the feature *positive_ nodes* when these features are selected on a branch.

Results Figure 5.3 shows learned trees (CART, unconstrained and constrained CPTrees) from this dataset. From this figure, it can be observed that both CART and the unconstrained CPTree violate the ordering constraint. Note that the constrained CPTree does satisfy the constraint but also its new top feature is the feature *age*, which appears to be, to some extents, more natural since it is likely to be the first question a medical doctor would ask to a breast-cancer patient. Additionally, Table 5.2 confirms that this prior knowledge is in phase with data since predictive accuracy is slightly improved for the constrained CPTree.

5.5.2 Must-be-selected Constraint Applied to Diabetes Prediction

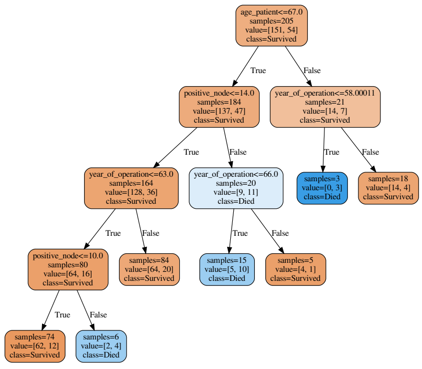
Here, we study the problem of predicting diabetes on patients of the dataset *Pima Indian diabetes* (Dua and Graff, 2017). Patients of this dataset are women of at least twenty one years old (Smith *et al.*, 1988). The dataset contains socio-demographic features such as age and clinical features such as body max index (BMI) and pregnancy-related features. In this use case, we impose the constraint that a feature related to the pregnancy (*pregnancy*) should be selected on the tree. Indeed, women who have been pregnant may have developed *gestational diabetes*, which may result in diabetes after giving birth (Read *et al.*, 2021).

Results Figure 5.4 shows learned trees from this diabetes dataset. According to the figure both CART and unconstrained CPTree select the same features but with different decision rules. They also fail to select a feature related to pregnancy. With these trees, it is difficult to quickly differentiate patients who are likely to have developed a gestational diabetes. In contrast, when infusing the prior information



(a) CART

(b) Unconstrained CPTree



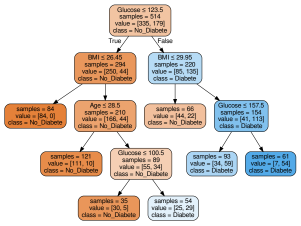
(c) Constrained CPTree

Figure 5.3: Ordering constraints. Decision trees obtained on Haberman's survival dataset using (a) CART, (b) CPTree without constraints and (c) CPTree with the constraint: "feature *age_patient* must appear before feature *positive_nodes*".

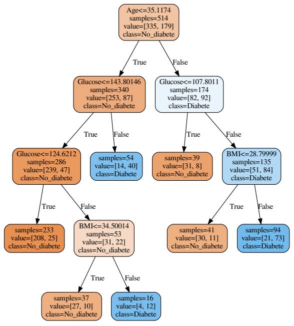
of the selection of the *pregnancy* feature the constrained CPTree provides decision rules where it may be possible to differentiate those patients. Furthermore, as shown in Table 5.2, this prior knowledge does not harm predictive accuracy, but helps to better generalise on unseen data.

5.5.3 Exclusion Constraint Applied to Prediction of Post-operative Action

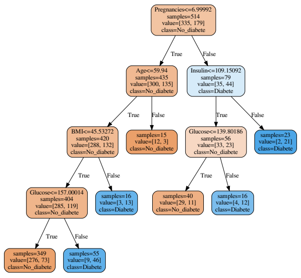
This application aims to predict whether a patient should stay in the same service, go to an intensive care unit or go back home for recovery after a surgery. We use



(a) CART



(b) Unconstrained CPTree



(c) Constrained CPTree

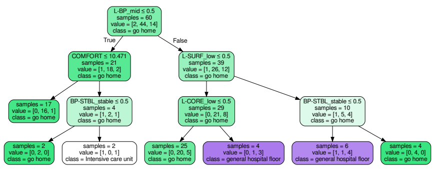
Figure 5.4: Must-be-selected constraints. Decision trees obtained on Diabetes dataset using (a) CART, (b) CPTree without constraints and (c) CPTree with the constraint: "feature *pregnancy* must appear on the tree".

	Train	**Test**
CART	83.33	53.33
CPTree	85.00	56.67
P-CART	80.00	63.63
C-CPTree	83.33	66.67

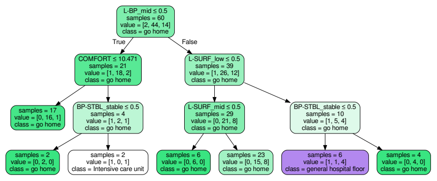
Table 5.3: Accuracy of CART, CPTree, P-CART (trained without the *L-CORE* feature) and constrained CPTree (C-CPTree) on Post- operative data.

the *Post-operative* dataset (Dua and Graff, 2017). Here, from prior knowledge, in average, the difference between the core temperature (*L-CORE*) and the surface temperature (*L-SURF*) is generally constant. So it would be surprising and useless to select these two (strongly correlated) features on the same branch.

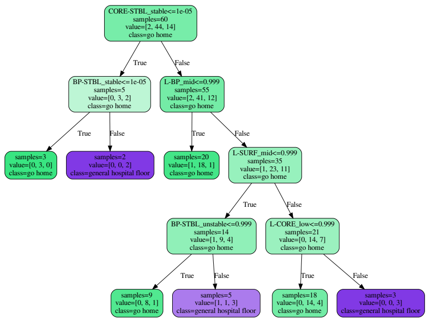
Results Table 5.3 shows performance of CART, CPTree, P-CART (CART with the feature *L-CORE* removed). It can be seen that adding this prior knowledge does not impair predictive accuracy. It further improves generalisation. Moreover, as shown in Figure 5.5, both CART and CPTree violate the constraint while with P-CART and C-CPTree, it is satisfied by design.



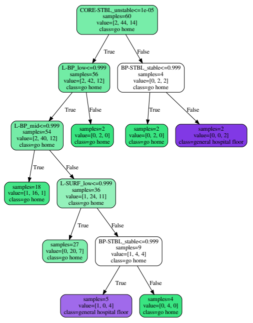
(a) CART



(b) CART on preprocessed data



(c) Unconstrained CPTree



(d) Constrained CPTree (C-CPTree).

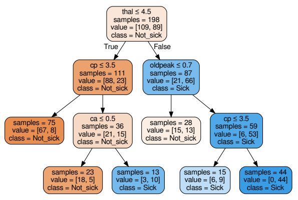
Figure 5.5: Exclusion constraints. Decision trees obtained on the Post-operative patient dataset with (a) CART without constraints, (b) CPTree without constraints, (c) CART with one of the two highly correlated features removed, (d) CPTree with the constraint: "*L-SURF* (surface temperature) and *L-CORE* (internal temperature) are mutually exclusive in a branch".

5.5.4 Minimum Accuracy on an Underrepresented Group and Test Cost Constraints Applied to Heart Disease Prediction

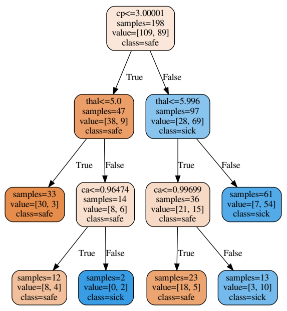
In this application, we use the *heart disease dataset* for heart disease prediction. In this application, usually, men have higher risks to develop a heart disease compared to women. Sick women represent therefore an underrepresented group in this domain application and very often they present atypical symptoms compared to men (Wenger *et al.*, 1993). Since even medical doctors have to be cautious (Okunrintemi *et al.*, 2018) when examining women patients for this disease, it is likely that a classifier will struggle to correctly classify these examples. In order to enhance possible trust of learned decision trees, we therefore impose the constraint that a high percentage (90%) of sick women should not be misclassified. We additionally impose the constraint on test costs, which is present in the paper **Nanfack** *et al.* (2022b).

	Train	**Test**	**Train Group**	**Test Group**
CART	82.32	77.78	68.75	55.56
CPTree	85.35	83.84	68.75	66.67
C-CPTree	81.82	79.80	93.75	88.89

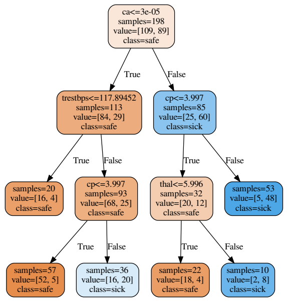
Table 5.4: Accuracy of CART, CPTree and constrained CPTree (C-CPTree) on heart disease. Group represents the group of sick women patients.



(a) CART



(b) CPTree



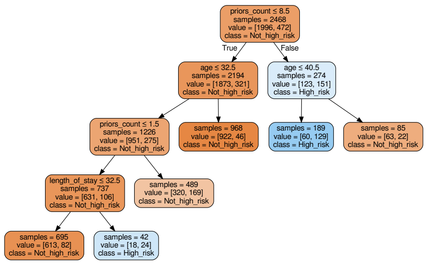
(c) Constrained CPTree

Figure 5.6: Minimum accuracy on a targeted group. Decision trees obtained on the heart disease dataset with (a) CART without constraints and (b) CPTree without constraint and (c) CPTree with the constraint : "90% of sick women should not be misclassified".

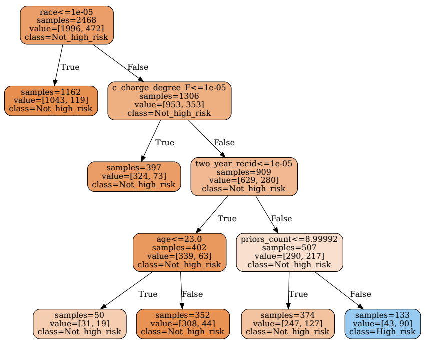
Results Table 5.4 shows results obtained on CART, unconstrained CPTree and the constrained CPTree. From this table, it can be observed that without the constraint, both CART and CPTree correctly classify only 68.75% of sick women. However, when imposing the constraint, the constrained CPTree does not only increase this percentage on the training distribution (93.75%), but also on the test distribution (88.89%). Nonetheless, in this case, the accuracy of constrained CPTree is slightly below the one of the unconstrained one. Learned trees can also be inspected in Figure 5.6.

	Train	**Test**	**Train DDP**	**Test DDP**
CART	83.60	83.76	8.09	7.49
CPTree	83.14	83.45	12.25	12.80
C-CPTree	81.64	82.18	1.68	1.37

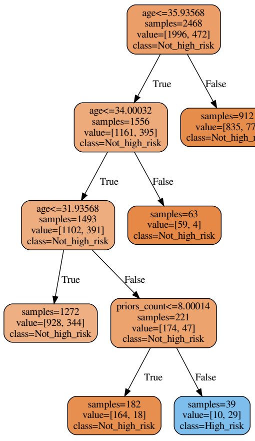
Table 5.5: Accuracy and DDP of CART, CPTree, constrained CPTree (C-CPTree) on Compas.



(a) CART



(b) Unconstrained CPTree



(c) Constrained CPTree

Figure 5.7: Fairness constraints. Decision trees obtained on COMPAS dataset using (a) CART, (b) CPTree without constraints and (c) CPTree with the constraint: "the DDP is less than 5%".

5.5.5 Fairness constraint Applied to Recidivism Prediction

In this application, we use the ProPublicas COMPAS recidivism data. The task is to predict a recidivism based on historical crime and demographic features. Studies have reported that the system built around this dataset was racially biased against African American defendants (Yang *et al.*, 2020). Inspired by Cotter *et al.* (2019), we impose the constraint that the DDP (with race as the protected feature) should be less than 5%.

Results Table 5.5 shows results obtained on CART, unconstrained CPTree and constrained CPTree. Trees can be visualised in Figure 5.7. From Table 5.5, it appears on one hand that, both CART and unconstrained have approximately the same predictive accuracy, but the unconstrained CPTree is less fair than CART according to DDP. On the other hand, when enforcing the fairness constraint DDP decreases by approximately at least 6% (down to $\pm 1\%$), while keeping the same level of accuracy of unconstrained trees.

In summary for these use-cases, this section showed that our framework allows enforcing domain-knowledge constraints in diverse real-world applications. For each application, our constrained CPTrees enforces constraints without loss in accuracy performance with respect to the unconstrained ones and CART baselines. In what follows, we show that if no constraints are enforced, CPTrees obtains competitive results with respect to state-of-the-art tree learners. We aim to show that domain experts can safely use our approach to obtain trees that (i) are reliable and (ii) straightforwardly enforce constraints that the models need to comply with.

5.6 Constraint-free Benchmarking

This section benchmarks the proposed CPTrees with respect to state-of-the-art decision tree learners in order to validate its relevance from an accuracy perspective. Indeed, Section 5.5 has shown that CPTrees can enforce domain knowledge constraints, but they must also provide competitive accuracy to be of practical interest.

5.6.1 Experimental Setting

Most of the experiment settings that we use in this section have been set in accordance with Verwer and Zhang (2019) and Bertsimas and Dunn (2017). Experiments have been performed on 20 UCI datasets (Dua and Graff, 2017) (mostly taken

Dataset name	N	M	Type of features	C
Balance scale	625	4	categorical	3
Bankote authentication	1372	5	numeric	2
Car evaluation	1728	6	categorical	3
Credit approval	690	15	categorical, numeric	2
Hepatitis	155	19	categorical, numeric	2
Ionosphere	351	34	numeric	2
Iris	150	4	numeric	3
Mammographic masses	961	6	categorical, numeric	2
Monk1	432	6	categorical	2
Monk2	432	6	categorical	2
Monk3	432	6	categorical	2
Pima indian diabetes	768	10	numeric	2
QSAR biodegradation	1055	41	numeric	2
Post operative patient	90	8	categorical, numeric	3
Seismic-bumps	2584	19	numeric	2
Spambase	4601	57	numeric	2
Spect heart	267	22	categorical	2
Thoracy surgery	470	17	numeric	2
Tic tac toe	958	9	categorical	2
Wine	178	13	numeric	3

Table 5.6: Datasets. N, M, C denote respectively the number of instances, features and classes.

from the list of datasets used by Verwer and Zhang (2019) plus additional ones as seen in Table 5.6). Depending on datasets, the number of classes varies from 2 to 3 and the number of instances from 90 to 4601.

No code for preprocessing datasets was found on any of the following source code repositories: BinOCT [4](Verwer and Zhang, 2019), Verhaeghe *et al.* (2019)[5] , OSDT[6] (Hu *et al.*, 2019) and DL8.5[7] (Aglin *et al.*, 2020). However, some datasets that we have used, were found on the BinOCT repository[4], were already preprocessed. In order to extend prepropreccessing for other datasets, we had to preprocess ourselves the datasets, which may explain little differences with published performances (e.g., slightly different accuracy) on few datasets. For methods that require binary features (DL8.5, OSDT, Verhaeghe *et al.* (2019) and our first model), datasets are preprocessed by transforming numeric features into

[4]https://github.com/SiccoVerwer/binoct

[5]https://bitbucket.org/helene_verhaeghe/classificationtree

[6]https://github.com/xiyanghu/OSDT

[7]https://github.com/aglingael/dl8.5

(3 bins using the quantile discretiser from Scikit-learn[8]) categorical ones and then transforming all categorical features into binary features through one-hot-encoding.

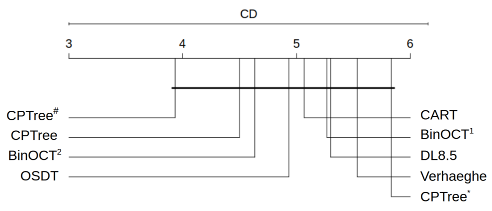
Datasets have been divided in 3 sets: training (50%), validation (25%) and testing (25%). BinOCT, DL8.5 and Verhaeghe *et al.* (2019) do not have any additional hyperparameters nor number of leaves to tune. We ran these models directly on training plus validation sets. Since BinOCT has an intrinsic heuristic to binarise numeric features, we keep datasets with their numeric features for experiments with BinOCT. This gives two versions of BinOCT (BinOCT^1 for only binary features and BinOCT^2 with numeric features). Other methods only work with binary features according to their released code. In the experiments, we also include CART (Breiman *et al.*, 1984) from Scikit-learn using the Shannon entropy as heuristic.

OSDT and CPTree require cross-validation to select hyperparameters λ and L, respectively. Therefore, λ has been selected after validation according to the default range of values provided by Hu *et al.* (2019). The number of leaves L of our model CPTrees has also been validated considering values from $3(K-1)$ to 2^K, where K is the maximum depth. Once tuning is done, we ran OSDT and our model on training plus validation sets, according to the best λ and L, respectively. Since our model is more general than Verwer and Zhang (2019), we made three versions of CPTrees. CPTree and CPTree* are respectively the model for complete tree structures ($L = 2^K$ as BinOCT) and the models for which the number of leaves has been validated. Second, CPTree* is the third model related to the second formulation (i.e., with numeric features) detailed in Section 5.3. CPTree and CPTree$^\#$ come from the first formulation (see Section 5.2).

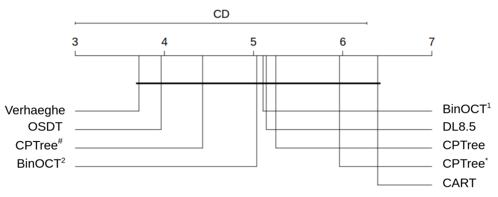
To be fair, all models ran without a warm start such as CART. No additional constraint has been added since the goal of this section is to assess the ability of our method to produce decision trees that achieve similar accuracy than those obtained by state-of-the-art methods described in Section 5.1.2.

Since our first formulation uses linear constraints and binary variables, it can be implemented into CP or MIP solvers. We used the CP-SAT solver of the Google OR-Tools library (Perron and Furnon, 2019), which is freely available in Python. For our second formulation, we used the Gurobi (Optimization, 2021) MIP solver. Regarding the optimisation of versions of CPTrees, there are several ways to improve the computational time (described in Section 5.8), e.g., by tuning technical parameters of solvers. We kept default ones and did not tune them to be fair as much as possible w.r.t. other tree learners.

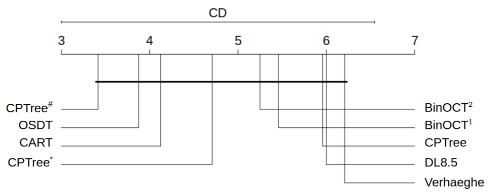
[8]We used KBinsDiscretizer from the preprocessing package.



(a) Maximum depth 2.



(b) Maximum depth 3.



(c) Maximum depth 4.

Figure 5.8: Nemenyi statistical significance test. From left to right, algorithms are ranked from best to worst. The bold horizontal line indicates no significant difference between algorithms.

Based on the work of BinOCT (Verwer and Zhang, 2019), we set to 10 minutes the time limit for each run of all the methods. Experiments have been conducted on 5 independent runs by dataset, by depth (the maximal depth was set to 2, 3 and 4) also according to Verwer and Zhang (2019). We ran them sequentially (to be fair with all methods) on a server with a Common KVM processor (2.294 GHz) and 16 GB of RAM. Running completely all the evaluations took more than 2 weeks[9] of extensive computations.

The code for CPTrees is available[10] and learned trees can be inspected and visualised as in Scikit-learn. All scripts that we used to benchmark all these recent models are also available for future reproducibility.

5.6.2 How do CPTrees Perform Comparatively to BinOCT, OSDT and DL8.5?

Our benchmarking procedure aims to evaluate the generalisation of optimal decision tree models rather than how close they are to optimal solutions. Therefore, Table 5.7a, 5.7b and 5.7c present the average test accuracy (on depth 2, 3 and 4, respectively) over 5 independent runs as detailed in Section 5.6.1. Similarly,

[9]The total number of runs is $\approx 20(\text{datasets}) \times 3(\text{depths}) \times 8(\text{models}) \times 5(\text{runs}) = 2400$.
[10]https://github.com/gerald4/CPTree

Dataset	CART	BinOCT^1	BinOCT^2	Verbakel *et al.* (2015)	DL8.5	OSDT	CPTree	CPTree$^\#$	CPTree*
Balance	61.15	63.06	65.35	N/A	65.35	N/A	65.35	66.50	66.62
Bank. A.	86.30	88.28	91.31	89.21	88.28	87.07	88.28	87.46	90.38
Biodeg	75.53	75.98	77.05	76.52	75.99	76.67	75.76	76.67	69.32
Car	77.13	77.13	77.13	N/A	77.13	N/A	77.13	77.13	72.08
Credit A.	85.37	84.27	84.63	85.37	84.03	85.37	84.27	85.61	74.27
Hepatitis	78.97	83.59	83.08	83.08	83.08	80.51	84.10	86.67	82.05
Ionosphere	78.64	81.82	88.86	81.82	81.82	78.18	82.27	80.45	89.55
Iris	93.68	93.68	90.53	N/A	94.21	N/A	93.68	93.68	92.11
Mam. M.	82.12	82.40	83.85	82.40	82.40	82.21	82.40	82.60	82.12
Monk1	76.83	75.02	75.02	74.55	75.02	74.59	75.97	75.54	75.97
Monk2	65.56	64.30	64.30	60.65	64.58	65.88	65.56	65.56	65.56
Monk3	95.83	95.70	95.70	95.96	95.70	95.70	95.83	95.83	95.83
Pima	74.58	74.58	74.38	74.58	74.58	73.75	74.58	74.58	73.02
Post O.	73.64	73.64	73.64	N/A	70.91	N/A	65.45	66.36	67.27
Seismic	93.44	93.34	93.10	93.03	93.34	93.44	93.34	93.34	93.19
Spambase	77.98	77.65	85.14	77.65	77.65	77.98	77.65	77.98	T/O
Spect H.	76.42	77.51	77.51	71.12	77.50	77.51	76.42	76.42	76.42
Thoracy S.	83.22	83.73	82.37	83.90	83.90	83.90	83.90	83.56	83.05
Tic T. T.	68.92	67.50	67.50	67.33	67.50	68.58	67.50	68.00	68.67
Wine	88.89	91.56	93.33	N/A	91.55	N/A	91.56	89.78	92.00

(a) Average test accuracy with maximum depth 2 over 5 repetitions.

Dataset	CART	BinOCT^1	BinOCT^2	Verbakel *et al.* (2015)	DL8.5	OSDT	CPTree	CPTree$^\#$	CPTree*
Balance	66.75	68.15	68.79	N/A	69.94	N/A	69.68	69.68	64.33
Bank. A.	86.30	92.54	96.15	94.46	92.54	92.30	92.59	90.50	92.13
Biodeg	76.67	78.41	78.33	82.58	80.08	80.53	79.09	80.30	66.67
Car	78.89	80.0	80.00	N/A	79.82	N/A	79.81	79.81	T/O
Credit	84.39	85.12	85.49	87.20	85.74	85.61	85.98	86.10	76.52
Hepatitis	79.49	81.03	80.51	82.05	81.54	80.51	81.03	81.03	82.56
Ionosphere	81.14	85.45	87.05	89.32	88.64	80.91	86.36	86.82	80.91
Iris	93.16	94.21	92.11	N/A	93.68	N/A	95.26	93.68	96.32
Mammo.	82.98	83.56	83.46	83.56	83.46	83.75	83.37	83.65	80.29
Monk1	80.43	86.27	86.27	86.70	86.27	85.16	85.18	80.86	82.45
Monk2	63.71	58.92	57.59	59.08	59.31	63.89	57.75	61.46	63.44
Monk3	98.42	99.14	99.14	99.28	96.36	98.58	98.99	98.99	97.41
Pima	73.96	72.71	73.65	70.52	70.52	73.75	70.62	73.02	71.48
Post O.	72.73	69.09	69.09	N/A	66.36	N/A	59.09	62.73	61.82
Seismic	93.44	93.28	93.13	93.28	93.19	93.44	93.22	93.19	93.34
Spambase	83.25	83.35	84.36	83.75	83.76	83.84	83.75	83.75	T/O
Spect H.	75.82	76.86	76.86	74.62	77.67	77.51	78.51	79.10	77.91
Thoracy S.	82.54	82.54	81.86	81.86	80.85	83.39	81.53	82.54	83.73
Tic T. T.	72.83	72.00	71.75	73.17	73.33	73.67	73.17	74.25	T/O
Wine	87.56	92.00	90.22	N/A	92.44	N/A	90.22	93.33	94.22

(b) Average test accuracy with maximum depth 3 over 5 repetitions.

Dataset	CART	BinOCT^1	BinOCT^2	Verbakel *et al.* (2015)	DL8.5	OSDT	CPTree	CPTree$^\#$	CPTree*
Balance	65.48	72.61	71.08	N/A	72.49	N/A	72.61	71.46	61.31
Bank.	92.36	93.94	97.26	95.63	94.58	92.77	94.46	93.76	93.41
Biodeg	77.42	78.79	79.09	81.06	80.38	78.56	78.26	79.17	T/O
Car	79.44	82.59	83.29	N/A	82.82	N/A	82.18	82.36	T/O
Credit A.	85.85	85.49	84.63	84.76	85.25	85.61	84.76	85.73	T/O
Hepatitis	78.97	74.87	82.56	78.46	78.46	76.92	82.56	83.08	82.05
Ionosphere	87.27	85.45	88.18	84.32	84.32	86.36	86.59	85.91	89.77
Iris	93.16	93.68	95.26	N/A	93.68	N/A	95.26	94.21	95.79
Mammo.	82.79	83.17	82.31	82.60	82.60	83.46	81.92	84.13	83.17
Monk1	82.88	100.00	100.00	100.00	100.00	100.00	100.00	100.00	81.58
Monk2	65.03	60.35	59.11	60.59	60.13	64.02	56.82	62.78	63.25
Monk3	98.99	97.47	97.47	96.19	97.28	98.58	98.56	98.99	96.83
Pima	71.88	71.77	71.25	69.27	70.52	72.92	69.38	72.08	74.35
Post O.	65.45	60.91	60.91	N/A	65.45	N/A	56.36	63.64	60.91
Seismic	93.44	93.13	92.94	93.13	93.06	93.44	93.07	93.16	93.30
Spambase	83.79	83.28	83.37	84.40	84.40	81.06	83.54	83.72	T/O
Spect H.	77.31	75.86	75.56	74.37	76.73	77.51	74.93	77.91	75.75
Thoracy S.	81.86	83.22	82.20	80.68	80.17	84.24	81.02	81.86	84.18
Tic T. T.	81.75	78.42	78.92	80.83	81.25	77.92	80.33	77.08	T/O
Wine	92.00	92.89	89.33	N/A	89.33	N/A	88.44	92.89	92.89

(c) Average test accuracy with maximum depth 4 over 5 repetitions.

Table 5.7: Test accuracy for several maximum depths. N/A means that the method cannot do multiclass classification. T/O means that no solution was found within the time limit (600s).

Dataset	CART	BinOCT^1	BinOCT^2	Verbakel *et al.* (2015)	DL8.5	OSDT	CPTree	CPTree$^\#$	CPTree*
Balance	65.64	69.87	69.10	N/A	69.10	N/A	69.10	68.72	68.68
Bank.	87.27	88.80	92.93	88.34	88.80	87.77	88.80	87.81	92.30
Biodeg	78.66	79.32	80.43	79.27	79.32	79.24	79.32	79.24	70.47
Car	77.99	77.99	77.99	N/A	77.99	N/A	77.99	77.99	71.40
Credit A.	86.95	87.28	87.53	86.71	87.28	86.71	87.28	87.03	75.01
Hepatitis	82.76	86.21	89.83	86.21	86.21	80.69	86.21	85.17	89.66
Ionosphere	80.99	86.01	91.94	86.01	86.01	83.88	86.01	85.32	90.42
Iris	95.18	95.54	92.86	N/A	95.54	N/A	95.54	95.18	93.93
Mammo.	83.18	84.08	84.66	84.08	84.08	83.47	84.08	83.89	84.12
Monk1	73.91	78.99	78.99	81.62	79.00	78.32	78.13	77.46	78.13
Monk2	65.78	65.88	65.88	76.80	65.88	65.05	65.78	65.78	65.78
Monk3	96.58	96.09	96.09	96.42	96.09	96.09	96.58	96.58	96.58
Pima	76.70	76.70	78.68	76.70	76.70	76.32	76.70	76.70	78.23
Post O.	72.62	75.38	75.38	N/A	80.00	N/A	75.38	73.85	75.38
Seismic	93.42	93.43	93.68	93.45	93.43	93.42	93.43	93.43	93.61
Spambase	78.67	78.99	85.43	78.99	78.99	78.67	78.99	78.67	T/O
Spect H.	80.00	79.35	79.35	85.12	79.35	79.35	80.00	80.00	80.00
Thoracy S.	86.08	86.48	86.82	86.48	86.48	85.85	86.48	86.14	86.42
Tic T. T.	70.75	71.23	71.23	71.23	71.23	70.81	71.23	71.17	69.67
Wine	93.08	93.83	97.29	N/A	93.83	N/A	93.83	93.53	96.99

(a) Average train accuracy with maximum depth 2 over 5 repetitions.

Dataset	CART	BinOCT^1	BinOCT^2	Verbakel *et al.* (2015)	DL8.5	OSDT	CPTree	CPTree$^\#$	CPTree*
Balance	70.34	75.00	74.87	N/A	75.00	N/A	75.09	73.80	66.32
Bank.	87.27	93.37	97.26	92.61	93.37	93.00	93.37	92.23	93.22
Biodeg	79.87	82.23	83.34	83.06	83.54	82.23	83.16	82.78	68.27
Car	79.63	81.20	81.20	N/A	81.57	N/A	81.57	81.57	T/O
Credit A.	87.28	88.71	88.63	89.16	89.57	86.99	89.57	88.63	78.53
Hepatitis	87.07	91.03	93.28	91.55	91.38	81.38	91.55	88.10	90.86
Ionosphere	85.40	92.02	93.92	93.16	93.08	85.32	92.17	90.57	86.84
Iris	95.54	98.39	99.64	N/A	98.22	N/A	98.39	96.79	98.75
Mammo.	84.50	85.14	85.34	85.31	85.27	83.92	85.31	84.57	78.91
Monk1	77.84	90.71	90.71	91.86	90.72	90.55	90.46	85.28	83.65
Monk2	66.09	69.71	69.76	73.59	69.71	65.63	68.62	67.47	66.71
Monk3	98.41	98.15	98.15	98.22	98.15	97.82	98.89	98.89	98.07
Pima	76.77	77.78	80.69	78.33	78.33	76.32	78.33	77.74	78.78
Post O.	75.38	81.85	81.85	N/A	84.92	N/A	82.15	77.54	77.54
Seismic	93.42	93.47	93.80	93.47	93.47	93.42	93.47	93.47	93.53
Spambase	84.08	83.86	84.61	84.22	84.22	84.22	84.22	84.22	T/O
Spect H.	81.00	82.25	82.25	84.29	82.00	79.35	82.20	82.10	81.90
Thoracy S.	86.76	87.44	87.90	88.12	88.01	86.02	88.12	86.82	86.31
Tic T. T.	75.96	77.30	77.19	78.80	78.50	76.77	78.77	76.96	T/O
Wine	95.19	97.89	99.85	N/A	97.89	N/A	97.89	96.39	99.40

(b) Average train accuracy with maximum depth 3 over 5 repetitions.

Dataset	CART	BinOCT^1	BinOCT^2	Verbakel *et al.* (2015)	DL8.5	OSDT	CPTree	CPTree$^\#$	CPTree*
Balance	72.01	78.21	77.95	N/A	79.06	N/A	78.93	76.79	61.75
Bank.	93.26	94.52	98.10	94.27	94.83	93.59	94.83	94.36	94.50
Biodeg	83.08	83.84	83.69	86.60	86.78	81.95	82.76	83.19	T/O
Car	80.19	83.63	83.53	N/A	84.55	N/A	83.89	83.80	T/O
Credit A.	89.98	89.61	89.08	91.41	91.74	88.02	90.67	89.53	T/O
Hepatitis	90.00	94.83	97.41	97.76	97.76	84.48	97.59	90.00	91.38
Ionosphere	90.11	94.52	95.59	97.26	97.26	89.05	95.59	92.02	91.33
Iris	97.50	98.39	100.00	N/A	98.39	N/A	98.39	97.68	97.68
Mammo.	85.14	85.92	86.11	86.43	86.40	84.21	86.21	85.24	81.94
Monk1	83.69	100.00	100.00	100.00	100.00	100.00	100.00	100.00	81.97
Monk2	68.53	73.93	73.83	77.78	74.51	69.40	72.22	70.13	66.89
Monk3	98.89	98.48	98.48	98.92	98.81	97.82	98.94	98.89	96.92
Pima	78.44	79.79	80.87	81.11	80.97	77.22	79.90	79.06	77.82
Post O.	77.85	86.46	86.77	N/A	91.69	N/A	90.46	79.08	78.77
Seismic	93.42	93.52	93.89	93.57	93.57	93.42	93.56	93.51	93.50
Spambase	84.60	84.84	84.01	85.50	85.50	81.88	84.57	84.52	T/O
Spect H.	82.90	85.95	85.95	88.54	86.95	79.35	86.90	84.60	82.25
Thoracy S.	88.01	88.81	88.86	90.28	90.17	85.51	89.77	88.12	86.08
Tic T. T.	83.62	83.82	83.87	87.05	86.69	81.78	84.43	80.84	T/O
Wine	98.80	99.10	100.00	N/A	100.00	N/A	100.00	97.44	98.80

(c) Average train accuracy with maximum depth 4 over 5 repetitions.

Table 5.8: Train accuracy obtained for several maximum depths. N/A means that the method cannot do multiclass classification. T/O means that no solution was found within the time limit (600s).

Table 5.8 shows the average training accuracy. Here, training accuracy of optimal methods differ because, as a reminder, in accordance with Verwer and Zhang (2019), all runs were done within the time limit of 10 minutes. Figure 5.8a, Figure 5.8b and Figure 5.8c show results of the Nemenyi statistical significant test, which is a non-parametric test that compares algorithms pairwise of their performance.

Figure 5.8 shows that none of the compared methods outperforms the others, in terms of generalisation, according to the Nemenyi test. This is confirmed by the analysis of Table 5.7, which shows predictive performance. In terms of test accuracy, Table 5.7 shows that OSDT and CPTree$^{\#}$ are usually close to each other, which is not surprising since they both are branch-like models. In terms of train accuracy, Table 5.8 shows that CPTrees are similar to state-of-the-art learners that are designed to find optimal decision trees.

It is also worth noting that, CPTree* is the only model (with CART) in Table 5.7 and Table 5.8, which does not involve heuristic discretisation of features. It is usually slow to train, in particular for datasets with categorical features. This is due to the existence of multiple choices of splits that gives the same semantic explanation. However, for pure numeric features (e.g., *Bank.*, *Ionosphere*, *Iris*, *Pima*, etc.), it usually provides good results, making it especially suitable for enforcing domain constraints in cases where datasets present numeric features.

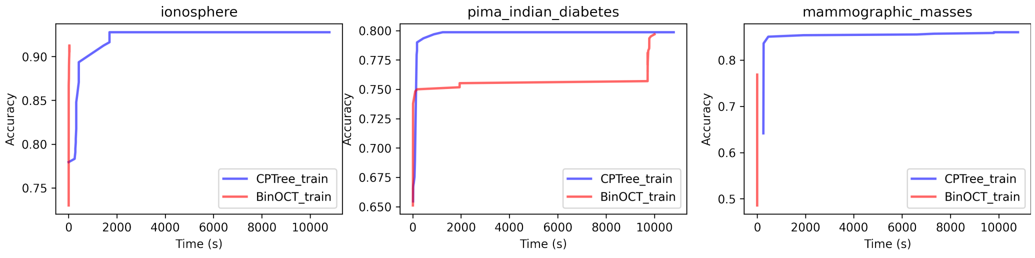
Overall, CPTree, CPTree$^{\#}$ and CPTree* generally have similar performances. They perform sometimes better (or worse) than state-of-the-art methods, but overall, there is no significant difference, according to the Nemenyi statistical tests. However, the particularity of CPTrees is to be as flexible as possible to incorporate a broad class of domain constraints as in Section 5.5.

5.7 On the Impact of Discretisation

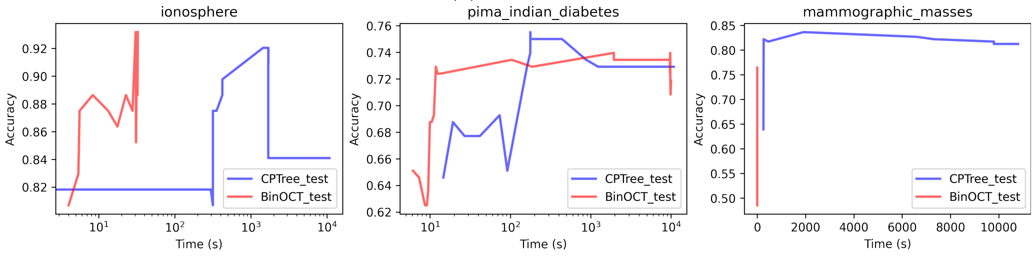
This section analyses the impact of discretisation for optimal tree learners that only work with binary features, namely the first version of CPTree, DL8.5 (Aglin *et al.*, 2020), BinOCT^{1} (Verwer and Zhang, 2019) and OSDT Hu *et al.* (2019).

On training data, the performance of tree learners such as BinOCT^{2} or CPTree* that work with continuous features is usually higher than the one of tree learners that only work with discretised binary features. When this performance is lower (e.g., Iris at depth 2 or credit approval at depth 3), either the considered method (BinOCT^{2} or CPTree*) has not reached the optimal tree (due to time limit) or the intrinsic heuristic of BinOCT^{2} used to binarise features is suboptimal.

On testing data, the performance of tree learners that work with continuous



(a) Training.



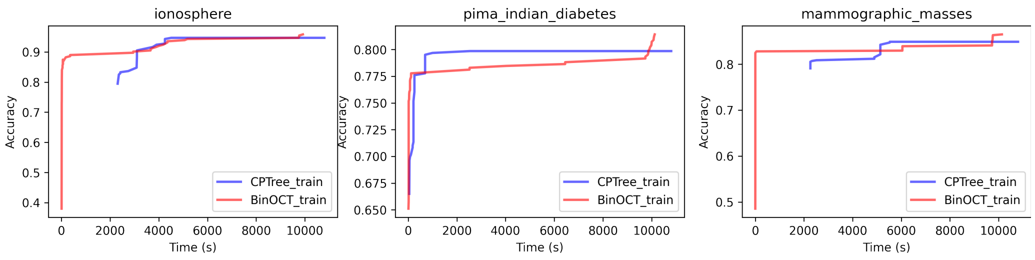
(b) Testing.

Figure 5.9: Training (a) and testing (b) accuracy curve within 3 hours of optimisation at the maximum depth of 3. This figure differs from the one in **Nanfack** *et al.* (2022b).

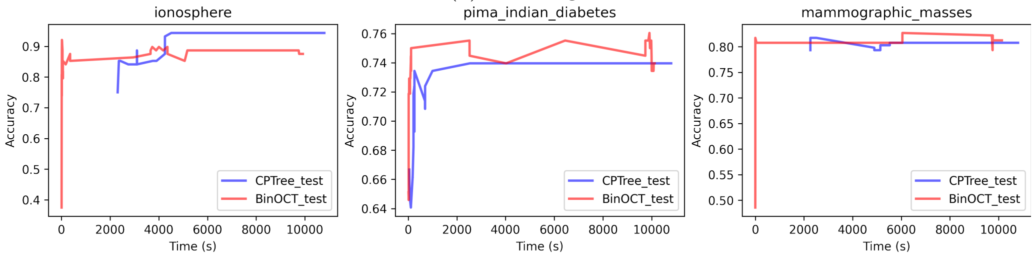
features is usually close to the one of tree learners that only work with binary features. When the former performance is higher than the latter one (up to $\approx 8\%$ for *wine* at depth 2), it means that the discretisation is detrimental for performance. When this performance is lower (e.g., *ionosphere* at depth 4), it means that this discretisation has played a kind of *simplicity bias*. It is important to note that the quantile discretiser we used is only a simple baseline. More advanced discretisers (Frank and Witten, 1999; Geurts and Wehenkel, 2000) for decision trees might be used with the aim to improve the predictive performance or the *stability* of decision trees.

5.8 Computational Time

This section examines the computational time required for CPTree to find a good decision tree that generalises well. This computational time is compared with the BinOCT method. We ran additional experiments on three datasets with maximum depths ($K = 3, 4; L = 2 * K$) and a time limit of 3 hours to explore more the feasible space. We implemented a *callback* to keep track of both the training and testing accuracy during the optimisation of CPTree and BinOCT.



(a) Training.



(b) Testing.

Figure 5.10: Training (a) and testing (b) accuracy curve within 3 hours of optimisation at the maximum depth of 4. This figure is a newly generated one and differs from the one in **Nanfack** *et al.* (2022b).

Figure 5.9 shows results obtained along optimisation path for depth 3. More specifically, Figure 5.9a shows that training accuracy saturates after a reasonable amount of time. This confirms that we can avoid increasing the time limit. However, from Figure 5.9b, the predictive accuracy monotonically increases and after a certain amount of time, it begins to fluctuate and sometimes decreases (with a non negligible drop for *Ionosphere* both CPTree and BinOCT). This supports the fact that oversearching the *optimal* tree do not always lead to improved generalisation.

Figure 5.9 shows similar results obtained along optimisation path at depth 4. The above analysis still hold, i.e., after a reasonable time training accuracy saturates while testing accuracy also saturates if not decreases. It is also important to note that in its formulation with numerical features, optimisation of CPTree is slower than BinOCT since the former has a wider search space than the latter.

In brief, our finding from this optimisation time/path analysis is that it is not necessary to reach optimality with CPTree (as well as BinOCT) since this may harm predictive performance without any strategies [11] to counter overfitting from oversearching. This oversearching problem is a well-known issue for optimal search-

[11]e.g., inductive biases through constraints

ing methods (Quinlan and Cameron-Jones, 1995). It needs further investigation especially on tree learners and we leave it for future work.

5.9 Conclusion

This chapter introduces a tree representation that leads to two new formulations to enforce domain-knowledge constraints on decision trees. With these formulations, we are able to enforce a broader family of constraints compared to recently proposed methods. These constraints include, but are not limited to the number of leaf nodes, the maximum depth, domain-knowledge constraints like the ordering of features on a branch of the tree, costs on features and even regarding fairness. These formulations provide a flexible framework in which several constraints both regarding the complexity and domain-knowledge can be easily formulated seeking to learn more interpretable and trustworthy trees. The learned CPTrees ensure that the constraints are satisfied while keeping the same level of accuracy with baselines. Future work includes more experiments to validate interpretability's improvement directly with users or domain experts.

Chapter 6

Constraint Enforcement for Global Explainability

In the two previous chapters, we were interested in imposing constraints on decision trees that are explainable by design. This chapter views explainability as soft constraints, which may need to be imposed on models that are not interpretable by design, i.e., black-box models. The chapter introduces a statistical framework in which differentiable black-box models are constrained to be easily explainable by decision trees. The chapter is largely based on the paper entitled "Global Explanations with Decision Rules: a Co-learning Framework" (**Nanfack** *et al.*, 2021b), and unless otherwise stated, all figures and tables come from that paper.

Contents

6.1 Context, Motivation, Problem and Related Work

This section gives an overview of the problem tackled in this chapter and presents the closely related work from the literature.

6.1.1 Context, Motivation and Problem

Black-box machine learning models can be extremely more accurate than interpretable models. Therefore, in several situations, machine learning practitioners may prefer using them. Yet, in critical applications, if models cannot be explained, domain experts will be reluctant to use them. More recently, special emphasis has been put on the need for machine learning models to provide explanations for their predictions in human-understandable terms (Doshi-Velez and Kim, 2017; Ribeiro *et al.*, 2016), in addition to accurate predictions. This research area is called eXplainable Artificial Intelligence (XAI). In XAI, on one hand, algorithms have been proposed to improve the performance of interpretable decision lists (Yang *et al.*, 2017), sets (Mita *et al.*, 2020) and trees (Verwer and Zhang, 2019). This corresponds to the interpretability or explainability by design. On the other hand, since in practice, more powerful models such as deep neural networks achieve impressive performances for tabular (Klambauer *et al.*, 2017), image (Chen *et al.*, 2020) and text data (Devlin *et al.*, 2019), external tools have been proposed to explain predictions of black-box models. This corresponds to *posthoc* explainability.

In posthoc explainability, two main families of explanations can be used: *global explanations* which explain *entirely* a complex model on its whole input space; and *local explanations* where an explanation is valid only in a specific region, which includes a particular instance (Guidotti *et al.*, 2018). This chapter focuses on global explanations of black-box models using decision rules (if-then rules), which are the most famous non-linear form of explanations (Lundberg *et al.*, 2020). Existing approaches for global explanations with decision rules (if-then rules) show some issues. For instance, the theoretical formalisation of *post-hoc* methods (Craven and Shavlik, 1995; Ribeiro *et al.*, 2018; Pedreschi *et al.*, 2019; Confalonieri *et al.*, 2020) is unclear (Craven and Shavlik, 1996; Wolf *et al.*, 2019) as well as whether their explanations reflect accurately the black-box model (Kim *et al.*, 2018; Slack *et al.*, 2020). The following section provides more details on what have been done to globally explain black-box models with decision rules.

6.1.2 Closely Related Work

To globally explain black-box models with rules, existing methods either are post-hoc or use prior knowledge.

6.1.3 Post-hoc Explainability Methods

Post-hoc explainability methods that use decision rules as explanations are generally called rule extraction methods. They consider a black-box model, and then learn an interpretable set, list or tree of decision rules to match its predictions. An early work is TREPAN (Craven and Shavlik, 1995), which approximates a neural network with a decision tree by learning m-of-n rules chosen to maximise the information gain ratio. There also exists a considerable literature of methods that use genetic algorithms (Boz, 2002; Arbatli and Akin, 1997), sampling strategies (Craven and Shavlik, 1994; Ribeiro *et al.*, 2018) and convex predicates (Gopinath *et al.*, 2019). However, their main limitation is that they are not stable (Melis and Jaakkola, 2018). In addition, there are no guarantees that explanations accurately reflect the knowledge captured by the complex black-box model (Kim *et al.*, 2018; Slack *et al.*, 2020).

6.1.4 Regularising for Explainability

Explaining black-box models with decision rules can also be done by regularising the black-box models. Two notable works are Okajima and Sadamasa (2019) and Wu *et al.* (2020). The former proposes to change the neural network architecture such that it can predict a rule (from a predefined rule set) and then a label given a particular instance. The latter leverages Wu *et al.* (2018) to enforce explainability by decision trees in local regions known *a priori*. In addition to being a challenging task (because of the discrete nature of rules), regularising for rule explanations with predefined rule sets or local regions has a major prerequisite. These rule sets or local regions are assumed to be known *a priori*. For explainability purposes, this is impractical, since they should be derived from the black-box model.

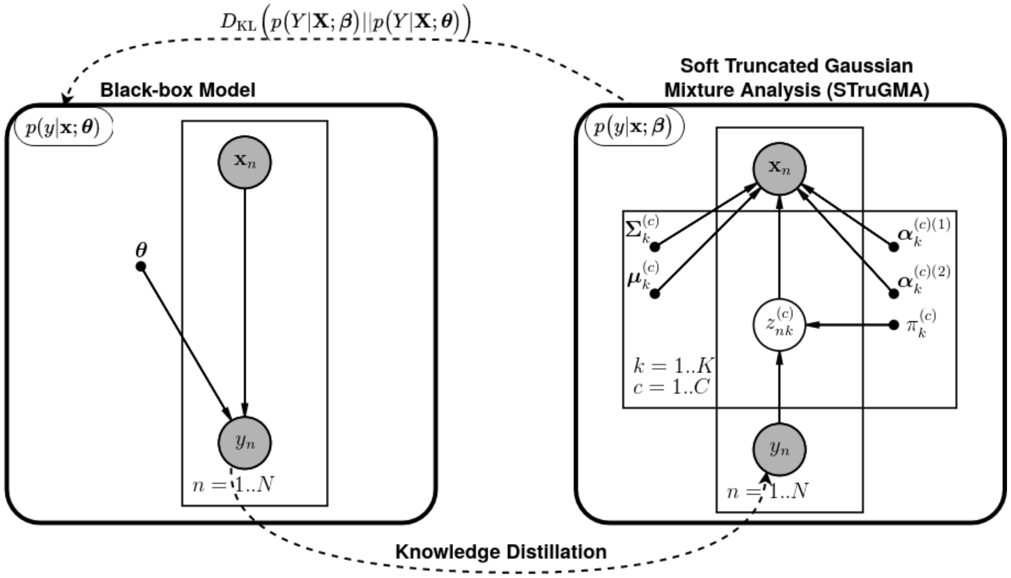

$D_{\mathrm{KL}}\left(p(Y|\mathbf{X};\boldsymbol{\beta})||p(Y|\mathbf{X};\boldsymbol{\theta})\right)$
Black-box Model
$p(y|\mathbf{x};\boldsymbol{\theta})$
$\mathbf{x}_n$
$\boldsymbol{\theta}$
y_n
$n = 1..N$
Soft Truncated Gaussian Mixture Analysis (STruGMA)
$p(y|\mathbf{x};\boldsymbol{\beta})$
$\mathbf{x}_n$
$\boldsymbol{\Sigma}_k^{(c)}$
$\boldsymbol{\mu}_k^{(c)}$
$\boldsymbol{\alpha}_k^{(c)(1)}$
$\boldsymbol{\alpha}_k^{(c)(2)}$
$z_{nk}^{(c)}$
$\pi_k^{(c)}$
$k = 1..K$
$c = 1..C$
y_n
$n = 1..N$
Knowledge Distillation

Figure 6.1: Co-learning between the black-box model (left) and STruGMA (right) through knowledge distillation and regularisation (dashed lines).

To address this problem, we propose a co-learning framework where hyper-rectangle rules are embedded into the newly introduced soft-truncated Gaussian mixture analysis (STruGMA). During co-learning (see Figure 6.1), STruGMA tries to explain the black-box model by the use of knowledge distillation, while the black-box model is learned with a regularisation with respect to STruGMA. This framework shares similarities with mutual distillation and posterior regularisation (Zhang *et al.*, 2018; Hu *et al.*, 2016).

6.2 Co-learning Framework To Enforce the Explainability Constraints

To embed decision rules in a differentiable *surrogate*, Section 6.2.1 proposes the soft truncated Gaussian mixture analysis (STruGMA), Section 6.2.2 proposes solutions for challenges that arise when learning STruGMA and Section 6.2.3 presents our co-learning strategy with the black box model.

6.2.1 STruGMA: Soft Truncated Gaussian Mixture Analysis for Differentiable Modelling

Geometrically, a rule defines a hyper-rectangle convex region $R(\boldsymbol{\alpha}_k) = \{\alpha_{kd}^{(1)} \leq x_d \leq \alpha_{kd}^{(2)}\}_{d=1}^{D}$, where $\alpha_{kd}^{(i)} \in \bar{\mathbb{R}}$ are the boundaries[1], D is the input space dimension and k is the index of the hyper-rectangle rule.

Motivated by the approximation properties of Gaussian distributions (thanks to the central limit theorem), we choose to map, as a surrogate, the k-th single rule to the truncated normal distribution

$$p\left(\boldsymbol{x}|z=k;\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(2)}\right) = \frac{\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\int_{\boldsymbol{\alpha}_k^{(1)}}^{\boldsymbol{\alpha}_k^{(2)}} \mathcal{N}(\boldsymbol{t};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)d\boldsymbol{t}} \quad \mathbb{1}\left\{\boldsymbol{\alpha}_k^{(1)} \leq \mathbf{x} \leq \boldsymbol{\alpha}_k^{(2)}\right\}(\boldsymbol{x}), \tag{6.1}$$

where $\mathbb{1}\{.\}$ is the indicator function and $\mathcal{N}(.;\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$ is the probability density function (pdf) of the multivariate normal (or Gaussian) distribution function with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. Optimising such a distribution is numerically unstable because of the piece-wise discontinuity of the indicator function $\mathbb{1}\{.\}$. However, the truncated normal distribution can be approximated by the soft truncated normal distribution (Souris *et al.*, 2018)

$$p\left(\boldsymbol{x}|z=k;\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k,\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(2)}\right) \approx \frac{\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\int_{\boldsymbol{\alpha}_k^{(1)}}^{\boldsymbol{\alpha}_k^{(2)}} \mathcal{N}(\boldsymbol{t};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)d\boldsymbol{t}} \prod_{d=1}^{D} \sigma_\eta\left(x_d - \alpha_{kd}^{(1)}\right) \tag{6.2}$$

$$\left(1 - \sigma_\eta\left(x_d - \alpha_{kd}^{(2)}\right)\right), \tag{6.3}$$

where $\sigma_\eta(x) = 1/(1+\exp(-\eta x))$ and η is a positive number. When $\eta \to +\infty, \sigma_\eta(x)$ tends towards $\mathbb{1}\{x \geq 0\}$. In practice, $\eta \geq 20$ is sufficient. Notice that this distribution reduces to the normal case when $\alpha_{kd}^{(1)} \to -\infty$ and $\alpha_{kd}^{(2)} \to +\infty$. Therefore, it can be interpreted as a normal distribution whose shape is constrained. Although its support is theoretically $\mathbb{R}$ in the univariate case, the high-density region is $[\alpha_{kd}^{(1)}, \alpha_{kd}^{(2)}]$. Figure 6.2 shows an example for $\mathbb{R}^2$ with $\eta = 20$.

Taking advantage of this distribution, we propose the (finite) soft truncated Gaussian mixture (STruGM) model to embed a set of hyper-rectangle rules in

[1]Note that the upper boundary $\alpha_{kd}^{(2)}$ can be $+\infty$ and the lower boundary $\alpha_{kd}^{(1)}$ can be $-\infty$, whenever relevant.

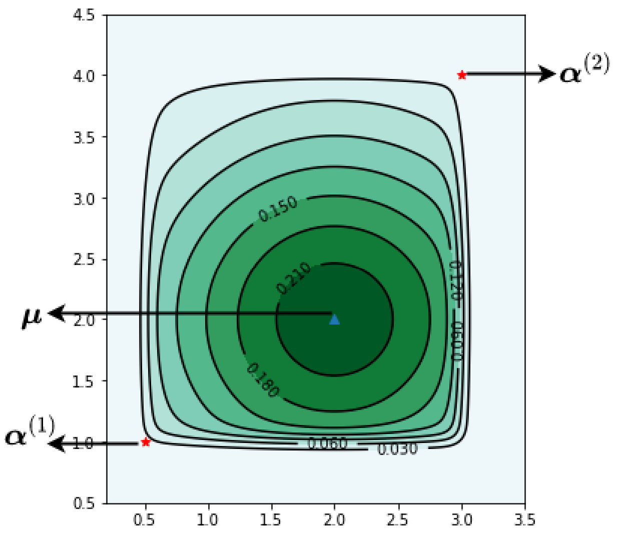


Figure 6.2: A soft truncated Normal Distribution.

a differentiable model. In addition, by drawing inspiration from the mixture discriminant analysis (MDA) (Hastie and Tibshirani, 1996), we propose the soft truncated Gaussian mixture analysis (STruGMA), i.e., a probabilistic generative classifier with class-conditional STruGM distributions. In STruGMA, each class has its own STruGM. In other words, STruGMA is a classifier $p(y|\mathbf{x};\boldsymbol{\beta})$ that reduces, when conditioning on the class, to the class-specific STruGM $p(\mathbf{x}|y;\boldsymbol{\beta}) = \sum_{k=1}^{K} p(z = k|y;\boldsymbol{\beta})p(\mathbf{x}|z = k, y;\boldsymbol{\beta})$, where K is the class-specific number of components and $\boldsymbol{\beta}$ are the parameters of STruGMA.

6.2.2 Adapting EM for STruGMA

Three challenges arise when learning STruGMA. Firstly, unlike the Gaussian distribution, it has been shown (Cohen Jr, 1950) that neither $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ nor $\boldsymbol{\alpha}$ have a closed-form solution for the maximum likelihood estimation (MLE) of a single truncated normal distribution. Secondly, learning the parameters $\boldsymbol{\alpha}^{(1)}$ and $\boldsymbol{\alpha}^{(2)}$ of STruGMA must satisfy the constraint $\boldsymbol{\alpha}^{(1)} < \boldsymbol{\alpha}^{(2)}$. Thirdly, learning STruGMA may result in many overlapping hyper-rectangle decision rules that are less interesting and more complex for explainability purposes (Fürnkranz *et al.*, 2012; Lakkaraju *et al.*, 2016).

STruGMA is a generative classifier with parameters $\boldsymbol{\beta} = \{\boldsymbol{\beta}^{(1)}, ..., \boldsymbol{\beta}^{(c)}, ..., \boldsymbol{\beta}^{(C)}\}$, where C is the number of classes and $\boldsymbol{\beta}^{(c)} = \{\boldsymbol{\pi}^{(c)}, \boldsymbol{\mu}^{(c)}, \boldsymbol{\alpha}^{(c)(1)}, \boldsymbol{\alpha}^{(c)(2)}, \boldsymbol{\Sigma}^{(c)}\}$.

$\boldsymbol{\alpha}^{(c)(1)}$ (resp. $\boldsymbol{\alpha}^{(c)(2)}$) $\in \mathbb{R}^{K_c \times D}$ is the lower (resp. upper) truncated point of the k-th component of class c; similarly, $\boldsymbol{\mu}^{(c)} \in \mathbb{R}^{K_c \times D}$ and $\boldsymbol{\Sigma}^{(c)} \in \mathbb{R}^{K_c \times D \times D}$. $\boldsymbol{\pi}^{(c)}$ are the mixing parameters and K_c is the number of components of class c. Here, $\boldsymbol{\Sigma}_k^{(c)}$ is diagonal for the sake of factorisation of the denominator. It can be easily extended by computing the multivariate Gaussian cumulative distribution function. Given these parameters, the joint distribution of STruGMA is

$$p(\mathbf{x}, y = c|\boldsymbol{\beta}) = p(y = c) \sum_{k=1}^{K_c} p\left(z^{(c)} = k|y = c; \boldsymbol{\pi}_k^{(c)}\right) \tag{6.4}$$

$$p\left(\mathbf{x}|z^{(c)} = k, y = c; \boldsymbol{\mu}_k^{(c)}, \boldsymbol{\Sigma}_k^{(c)}, \boldsymbol{\alpha}_k^{(c)(1)}, \boldsymbol{\alpha}_k^{(c)(2)}\right) \tag{6.5}$$

$$= p(y = c) \sum_{k=1}^{K_c} \boldsymbol{\pi}_k^{(c)} \frac{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k^{(c)}, \boldsymbol{\Sigma}_k^{(c)})}{\int_{\boldsymbol{\alpha}_k^{(c)(1)}}^{\boldsymbol{\alpha}_k^{(c)(2)}} \mathcal{N}(\mathbf{t}; \boldsymbol{\mu}_k^{(c)}, \boldsymbol{\Sigma}_k^{(c)}) d\mathbf{t}} \tag{6.6}$$

$$\prod_{d=1}^{D} \sigma_\eta\left(x_d - \alpha_{kd}^{(c)(1)}\right)\left(1 - \sigma_\eta\left(x_d - \alpha_{kd}^{(c)(2)}\right)\right). \tag{6.7}$$

From now, for simplicity, the class conditioning c is omitted for parameters. One of the most popular method to learn finite mixture models is the expectation-maximisation (EM) algorithm, described in Section 2.3.2. As it consists of a mixture per class, STruGMA can be learned by adapting EM. With the parameters $\boldsymbol{\beta} = \{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)}\}$, EM maximises the expected log-likelihood $Q(\boldsymbol{\beta}, \boldsymbol{\beta}^t)$ by alternating the following steps:

- E-step: computing class responsibilities

$$r_{nk} = p\left(z = k|\boldsymbol{x}_n; \boldsymbol{\beta}^t\right) = \frac{\pi_k p(\boldsymbol{x}_n|z = k; \boldsymbol{\beta}^t)}{\sum_{k_1} \pi_{k_1} p(\boldsymbol{x}_n|z = k_1; \boldsymbol{\beta}^t)}; \tag{6.8}$$

- M-step: because of the lack of closed-form solution of parameters through the MLE, the M-step performs a gradient descent on the negative expected loglikelihood

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t + \epsilon_t \nabla Q\left(\boldsymbol{\beta}, \boldsymbol{\beta}^t\right), \tag{6.9}$$

where ϵ_t is the learning rate and

$$Q(\boldsymbol{\beta}, \boldsymbol{\beta}^t) = \sum_n \sum_k r_{nk} \log \pi_k + \sum_n \sum_k r_{ik} \Big[\log \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) +$$

$$\sum_d \log \sigma_\eta \left(x_{nd} - \alpha_{kd}^{(1)}\right) + \log \left(1 - \sigma_\eta \left(x_{nd} - \alpha_{kd}^{(2)}\right)\right) \Bigg]$$

$$- \sum_n \sum_k r_{nk} \log \int_{\boldsymbol{\alpha}_k^{(1)}}^{\boldsymbol{\alpha}_k^{(2)}} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) d\mathbf{x}. \quad (6.10)$$

Details about gradients $\nabla Q(\boldsymbol{\beta}, \boldsymbol{\beta}^t)$ are given in the supplementary material of the paper **Nanfack** *et al.* (2021b).

Directly optimising with gradient descent in the M-Step, as described, may raise difficulties caused by the definition of the denominator of the soft truncated normal distribution. Indeed, we need to impose $\boldsymbol{\alpha}^{(2)} > \boldsymbol{\alpha}^{(1)}$ as a hard constraint. To solve the problem, we leverage the projected gradient descent method (See Section 2.5.2.3) on the constraint set $S = \{\boldsymbol{\alpha}^2 > \boldsymbol{\alpha}^1\}$. The projected gradient solves the problem

$$\boldsymbol{\alpha}^{t+1} = \text{Proj}_{\boldsymbol{\alpha} \in S}(\boldsymbol{\alpha}^t + \epsilon_t \nabla Q(\boldsymbol{\alpha}, \boldsymbol{\alpha}^t)) \quad (6.11)$$

$$= \text{argmin}_{\boldsymbol{\alpha}} ||\boldsymbol{\alpha} - (\boldsymbol{\alpha}^t + \epsilon_t \nabla Q(\boldsymbol{\alpha}, \boldsymbol{\alpha}^t))|| \quad (6.12)$$

$$\text{s.t. } \boldsymbol{\alpha}^{(2)} > \boldsymbol{\alpha}^{(1)} \quad (6.13)$$

and, using the K.K.T. conditions (provided in 2.5.2.3) on this constrained quadratic optimisation problem, one obtains

$$\begin{cases} \boldsymbol{\alpha}_{t+1}^{(1)} = \boldsymbol{\alpha}_t^{(1)} + \epsilon_t \nabla Q(\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}_t^{(1)}) \\ \boldsymbol{\alpha}_{t+1}^{(2)} = \boldsymbol{\alpha}_t^{(2)} + \epsilon_t \nabla Q(\boldsymbol{\alpha}^{(2)}, \boldsymbol{\alpha}_t^{(2)}) \end{cases}$$

if $\boldsymbol{\alpha}_{t+1} \in S$ and, otherwise,

$$\begin{cases} \boldsymbol{\alpha}_{t+1}^{(1)} = \frac{1}{2}\Big(\boldsymbol{\alpha}_t^{(1)} + \boldsymbol{\alpha}_t^{(2)} \\ \qquad + \epsilon_t \left(\nabla Q(\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}_t^{(1)}) + \nabla Q(\boldsymbol{\alpha}^{(2)}, \boldsymbol{\alpha}_t^{(2)})\right) - \zeta\Big) \\ \boldsymbol{\alpha}_{t+1}^{(2)} = \frac{1}{2}\Big(\boldsymbol{\alpha}_t^{(1)} + \boldsymbol{\alpha}_t^{(2)} \\ \qquad + \epsilon_t \left(\nabla Q(\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}_t^{(1)}) + \nabla Q(\boldsymbol{\alpha}^{(2)}, \boldsymbol{\alpha}_t^{(2)})\right) + \zeta\Big) \end{cases}$$

The margin $\zeta > 0$ is a small number used to transform the strict inequality into inequality constraint $\boldsymbol{\alpha}^{(2)} \geq \boldsymbol{\alpha}^{(1)} + \zeta$.

For complexity and explainability purposes, it is useful to have non-overlapping hyper-rectangle rules. For two hyper-rectangle rules i and j, this is formalised as (Xu *et al.*, 2019)

$$\max_d \left(\left| \frac{1}{2} \left(\alpha_{id}^{(1)} + \alpha_{id}^{(2)} \right) - \frac{1}{2} \left(\alpha_{jd}^{(1)} + \alpha_{jd}^{(2)} \right) \right| \right.$$
$$\left. - \frac{1}{2} \left(\alpha_{id}^{(2)} - \alpha_{id}^{(1)} \right) - \frac{1}{2} \left(\alpha_{jd}^{(2)} - \alpha_{jd}^{(1)} \right) \right) \geq 0.$$

Enforcing this constraint is a difficult problem in the literature. We tackle it with a simple, yet effective heuristic. Based on the form of the constraint (the maximum is positive when only one of the values is positive), it consists in choosing a specific dimension d and adapting either $\boldsymbol{\alpha}_i$ or $\boldsymbol{\alpha}_j$ along d to satisfy the constraint. The corresponding choices are taken to maximise the expected log-likelihood. Details are discussed in the supplementary material of the paper **Nanfack** *et al.* (2021b).

6.2.3 Co-learning STruGMA and Black-box Models for Rule Explanations

This section proposes a co-learning framework where (i) hyper-rectangle rules of STruGMA are learned to globally explain a black-box model and (ii) this black-box is simultaneously constrained by STruGMA to be easier to explain.

6.2.3.1 Co-learning of the Black-box Model

Let us consider a probabilistic black-box model $p(y|\mathbf{x};\theta)$ that can be trained with gradient descent. Our goal is to constrain it to follow hyper-rectangle rules of STruGMA as much as possible. This is achieved by using the loss

$$\lambda \times \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) + (1-\lambda) \times D_{\mathrm{KL}}\big(p(\mathbf{Y}|\mathbf{X};\boldsymbol{\beta})||p(\mathbf{Y}|\mathbf{X};\boldsymbol{\theta})\big), \tag{6.14}$$

where $\lambda \in [0,1]$ can be a hyper-parameter, $\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta})$ is a usual loss on training data such as the cross-entropy, D_{KL} is the Kullback–Leibler divergence between the reference model $p(\mathbf{Y}|\mathbf{X};\boldsymbol{\beta})$ given by STruGMA and the black-box model $p(\mathbf{Y}|\mathbf{X};\boldsymbol{\theta})$ which is optimised. This divergence acts as a regularisation term that encourages the black-box model to satisfy hyper-rectangle rules of STruGMA. It is a conditional expectation and can be evaluated as

$$D_{\mathrm{KL}}\big(p(\mathbf{Y}|\mathbf{X};\boldsymbol{\beta})||p(\mathbf{Y}|\mathbf{X};\boldsymbol{\theta})\big) = \mathbb{E}_{\boldsymbol{x}\sim p(\mathbf{x};\boldsymbol{\beta})}[D_{\mathrm{KL}}(p(\mathbf{Y}|\boldsymbol{x};\boldsymbol{\beta})||p(\mathbf{Y}|\boldsymbol{x};\boldsymbol{\theta}))] \tag{6.15}$$
$$\approx \frac{1}{N_s} \sum_{i=1}^{N_s} \sum_{c=1}^{C} p(y=c|\hat{\boldsymbol{x}}_i;\boldsymbol{\beta}) \log \frac{p(y=c|\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})}{p(y=c|\hat{\boldsymbol{x}}_i;\boldsymbol{\theta})}, \tag{6.16}$$

where $\{\hat{\boldsymbol{x}}_i\}_{i=1}^{N_s}$ is a new sample obtained from STruGMA to compute a Monte-Carlo estimate of the divergence term. Sampling from STruGMA has a complexity which is linear with respect to the input space dimension D.

One issue regarding the performance of the learned black-box model, after optimising Eq. 6.14, is its sensitivity with respect to the choice of λ. Indeed, it can result in a (too) weakly or strongly constrained black-box model. This problem is ubiquitous in multi-objective optimisation. To alleviate this sensible choice of λ, we apply the multiple gradient descent algorithm (MGDA)(Sener and Koltun, 2018; Désidéri, 2009), which consists in finding, at each iteration, the λ^* that gives the direction of gradient that improves both terms of Eq. 6.14. This λ^* is the one that minimises $||\lambda\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) + (1-\lambda)\nabla_{\boldsymbol{\theta}} g(\boldsymbol{\beta}, \boldsymbol{\theta})||$ and is obtained using

$$\lambda^* = \left[\frac{\left(\nabla_{\boldsymbol{\theta}} g(\boldsymbol{\beta},\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})\right)^\top \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\beta},\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\beta},\boldsymbol{\theta})\|_2^2}\right]_{+,\frac{1}{T}}, \tag{6.17}$$

where $f(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta})$, $g(\boldsymbol{\beta},\boldsymbol{\theta}) = \hat{D_{\mathrm{KL}}}\big(p(\mathbf{Y}|\mathbf{X};\boldsymbol{\beta})||p(\mathbf{Y}|\mathbf{X};\boldsymbol{\theta})\big)$, and $[\cdot]_{+,\frac{1}{T}} = \max(\min(.,1),0)$ is a clipping operation to $[0,1]$.

Algorithm 6.1 Co-learning of a black-box model with STruGMA

```
Input: training set $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$, black-box model with parameters $\boldsymbol{\theta}$, number of
    epochs $N_{\text{epochs}}$, number of gradient descent steps in the M-step $N_{\text{STruGMA}}$,
    number of rules $K$ per class, size of MC sample $N_s$
Output: trained black-box model and STruGMA
 1: initialise STruGMA with GMMs per class
 2: while not converged do
 3:     // update black-box model (regularised GD)
 4:     draw an MC sample $\{\hat{\boldsymbol{x}}_i\}_{i=1}^{N_s}$ from STruGMA
 5:     get $\lambda*$ from Eq. 6.17 and train the black-box model with $\lambda^*\mathcal{L}(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{\theta})$ +
    $(1-\lambda^*)\hat{D_{\mathrm{KL}}}(p(\mathbf{Y}|\mathbf{X};\boldsymbol{\beta})||p(\mathbf{Y}|\mathbf{X};\boldsymbol{\theta}))$ for $N_{\text{epochs}}$ epochs

 6:     // update STruGMA (knowledge distillation)
 7:     relabel training set with black-box model
 8:     E-step of STruGMA with Eq. 6.8 (responsabilities)
 9:     M-step of STruGMA with Eq. 6.9 ($N_{\text{STruGMA}}$ iterations of gradient descent
    + gradient projection)
10: end while
11: return STruGMA and the black box
```

Table 6.1: Details of the datasets for the experiments.

Dataset	Size	Dimension
Wine	178	13
Pima Indian diabetes (Pima)	768	8
Ionosphere	351	34
Magic gamma (Gamma)	19020	11
Bank marketing (Marketing)	4119	20
German credit (Credit)	1000	20
Waveform	5000	40

6.2.3.2 Co-learning of STruGMA through Knowledge Distillation

As we want STruGMA to globally explain the black-box model with hyper-rectangle decision rules, one approach is to use knowledge distillation. Training instances $\boldsymbol{X}$ are relabelled with the outputs $\boldsymbol{Y_\theta}$ of the black-box model and STruGMA is learned from this new dataset. As a result, STruGMA approximates the black-box model on the whole input space. The resulting co-learning Algorithm 6.1 (see also Figure 6.1) summarises how to learn both models. It has the key advantage to work with any black-box model that is learned through gradient-based optimisation.

6.3 Empirical Results and Discussion

Experiments assess whether (i) after co-learning with STruGMA the black-box's decision boundary becomes easier to approximate by a rule learner with a limited impact on its accuracy, (ii) decision rules explanations from distilled decision trees after co-learning are more faithful than those without co-learning. In the paper **Nanfack** *et al.* (2021b), we also performed a qualitative evaluation to verify whether extracted rules comply with domain knowledge.

6.3.1 Experimental Settings

We validate our method [2] on a synthetic dataset and on seven commonly used machine learning datasets from UCI (Dua and Graff, 2017) for which neural networks usually outperform decision trees; see Table 6.1 for their details. We chose deep neural networks with architectures inspired from the literature (Wu

[2]Available at `https://github.com/gerald4/Co-learning_with_STruGMA`.

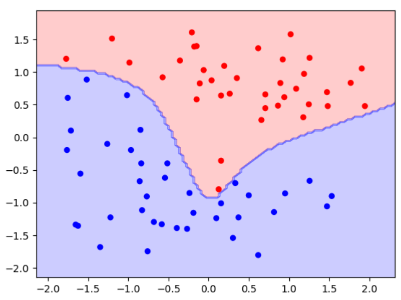
et al., 2020; Ribeiro *et al.*, 2018; Pedapati *et al.*, 2020), as they work well for tabular data. Our dense hidden layers have *ELUs* as activation functions and the last layer has the dimension of the number of classes. Details about these architectures are left in the supplementary material. At each iteration of the co-learning, only $N_{\text{epochs}} = 1$ epochs are spent for updating the MLP. Indeed, if the complexity of the black-box model changes too rapidly, STruGMA will not be able to explain it and the D_{KL} term will not be able to play its role as a regulariser. The number of iterations $N_{\text{STruGMA}} = 100$ for the M-step ensures that STruGMA is a good approximation of the black-box model at each iteration. The size of the Monte Carlo sample to approximate the divergence between the two models is set to $N_s = 10 \times N$. The margin ζ between $\boldsymbol{\alpha}^{(1)}$ and $\boldsymbol{\alpha}^{(2)}$ is set to a relatively small value of 0.2. A Gaussian mixture model is used to initialise STruGMA, with $\boldsymbol{\alpha}_k^{(1)} = \boldsymbol{\mu}_k - 0.2\boldsymbol{\sigma}_k$ and $\boldsymbol{\alpha}_k^{(2)} = \boldsymbol{\mu}_k + 0.4\boldsymbol{\sigma}_k$ for each component. The number of components K is the same for each STruGM per class and is chosen in $\{2, 3, 4\}$ to avoid complex rule explanations. This hyper-parameter is chosen with a separate validation set. Both STruGMA and the black-box model are learned using the gradient-based optimiser Adam (Kingma and Ba, 2015) with 10^{-3} as learning rate.

We use the accuracy (percentage of correct predictions) to assess the quality of predictions and the fidelity (percentage of predictions where a black-box and a white-box model agree) to measure the mutual agreement.

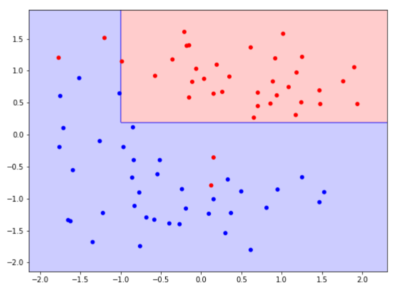
It is possible to directly use hyper-rectangle rules of STruGMA as explanations when the input space dimension D is relatively low. However, to get simple rules with limited size, we resort to decision trees as rule learners to provide global explanations. When co-learning is done, similarly as post-hoc methods, a *distilled* decision tree is trained by mimicking predictions of the co-learned black-box model. We explore two possibilities: either we use the predictions of the co-learned black-box model with original features to train a distilled decision tree (TreeCoExplainerBB) or we use the predictions of the co-learned black-box model with hyper-rectangle splits transformed as binary features to train the distilled decision tree (TreeCoExplainerHR). Both are compared with a baseline (TreeExplainer) where the distilled decision tree is trained to mimic the black-box model without co-learning. This baseline is representational of the global post-hoc (through decision trees) explanation methods discussed in Section 6.1.3.

6.3.2 Effect of Co-learning on Model Inference

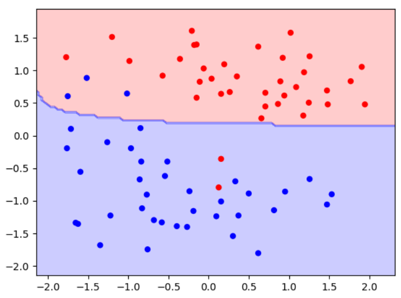
Figure 6.3 illustrates the effect of co-learning on a synthetic two-dimensional toy example. Figure 6.3a shows a black-box model learned without any kind of co-



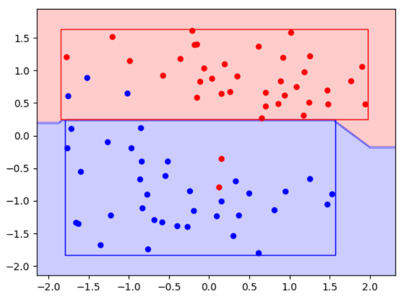
(a) Two-layer MLP without co-learning



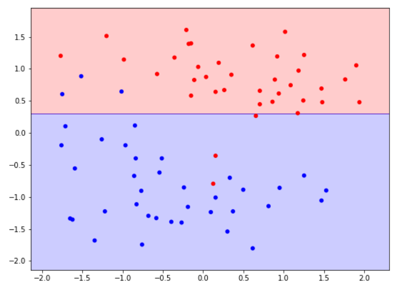
(b) TreeExplainer for the MLP in Fig. 6.3a



(c) Two-layer MLP co-learned with STruGMA



(d) STruGMA co-learned with MLP of Fig. 6.3c



(e) TreeCoExplainerHR with STruGMA splits

Figure 6.3: Decision boundary of several models, (a-b) without and (c-e) with co-learning, on a tw o-dimensional example. The black box model is easier to explain with rules after co-learning (c) than in the standard case (a).

Table 6.2: Impact of co-learning on the test fidelity. Mean and standard deviation over 10 repetitions are reported. TreeExplainer is the distilled decision tree obtained with the predictions of the black-box model without co-learning, whereas TreeCoExplainerHR (resp. TreeCoExplainerBB) is our distilled decision tree of the co-learned black-box model using the hyper-rectangle rules of STruGMA as binary features (resp. original features).

Dataset	**TreeExplainer**	**TreeCoExplainerHR**	**TreeCoExplainerBB**
Bank	95.97 (0.74)	96.18 (0.63)	**96.49 (0.89)**
Credit	77.3 (3.47)	81.25 (3.47)	**81.5 (3.43)**
Ionosphere	87.32 (3.25)	**90.28 (3.42)**	88.87 (5.69)
Gamma	93.31 (2.08)	93.15 (0.85)	**95.6 (0.36)**
Pima	88.44 (2.41)	88.9 (1.35)	**92.01 (3.24)**
Waveform	80.26 (1.53)	80.52 (1.87)	**80.86 (1.28)**
Wine	89.17 (4.62)	**92.78 (4.93)**	89.72 (2.64)

learning, returning a decision boundary which is somewhat complex for a simple toy problem. A distilled decision tree is then learned in Figure 6.3b to explain this black-box model. Although the region where $x_1 < -1$ and $x_2 > 1.1$ contains training instances, the decision tree fails to explain it correctly. This problem is avoided with the co-learning of the black-box model in Figure 6.3c and STruGMA in Figure 6.3d. The corresponding distilled decision tree (our TreeCoExplainerHR) in Figure 6.3e globally explains the co-learned black-box model. This toy example illustrates how co-learning rectifies the decision boundary of a black-box model to be compatible with rules. The co-learned black-box model is very likely to follow rule explanations extracted by rule learners such as decision trees. Note that the black-box model in Figure 6.3a may be more accurate than its co-learned version in Figure 6.3c.

6.3.3 Impact on Fidelity and Accuracy

Table 6.2 shows the test fidelity of the baseline TreeExplainer and our methods TreeCoExplainerHR and TreeCoExplainerBB. Tree depth has been cross-validated. On all datasets, results show that co-learning improves fidelity between the black-box model and distilled decision trees. This means that one can be more confident in explanations based on decision trees after co-learning with STruGMA than without co-learning.

Table 6.3: Predictive accuracy of co-learned black-box models (coBB) and black-box models without co-learning (BB). Mean and standard deviation are shown over 10 repetitions.

Dataset	coBB	BB
Bank	90.68 (0.77)	**90.99 (0.84)**
Credit	**75.65 (3.88)**	74.75 (3.5)
Ionosphere	**90.98 (3.88)**	90.56 (3.45)
Gamma	80.57 (0.49)	**82.79 (2.53)**
Pima	73.12 (2.31)	**75.39 (1.77)**
Waveform	85.97 (0.87)	**86.15 (0.7)**
Wine	96.94 (2.43)	**97.5 (2.05)**

Table 6.4: Predictive accuracy of distilled trees. Mean and standard deviation over 10 repetitions are reported.

Dataset	TreeExplainer	TreeCoExplainerHR	TreeCoExplainerBB
Bank	**91.29 (0.94)**	90.42 (1.0)	90.81 (0.99)
Credit	69.15 (3.33)	71.5 (2.59)	**71.55 (4.7)**
Ionosphere	**88.03 (3.89)**	87.18 (4.01)	86.34 (3.45)
Gamma	**80.79 (2.09)**	77.04 (1.12)	79.16 (0.37)
Pima	**72.4 (1.44)**	71.24 (3.17)	71.88 (2.12)
Waveform	76.43 (1.9)	76.38 (1.83)	**76.71 (1.58)**
Wine	89.17 (3.33)	**91.67 (4.54)**	88.89 (3.93)

Table 6.3 shows the accuracy of the black-box model (BB) without co-learning and the co-learned black-box model (coBB). It can be seen that the co-learning usually negatively impacts the test accuracy of the black-box model (except on *Credit* and *Ionosphere*). However, the difference is usually not important, as it is usually around 2%. This difference is also perceived on the accuracy of distilled decision trees in Table 6.4. Nonetheless, as it can be seen in Table 6.5, our co-learned black-box models still usually perform better than an interpretable decision tree.

Overall, in addition to providing faithful global explanations thanks to co-

Table 6.5: Predictive accuracy of co-learned black-box models (coBB) and decision trees (DT). Mean and standard deviation over 10 repetitions are reported.

Dataset	**coBB**	**DT**
Bank	90.68 (0.77)	**90.81 (0.96)**
Credit	**75.65 (3.88)**	71.05 (3.3)
Ionosphere	**90.98 (3.88)**	90.28 (4.43)
Gamma	80.57 (0.49)	**82.72 (0.43)**
Pima	**73.12 (2.31)**	72.02 (2.59)
Waveform	**85.97 (0.87)**	75.24 (1.23)
Wine	**96.94 (2.43)**	87.78 (4.93)

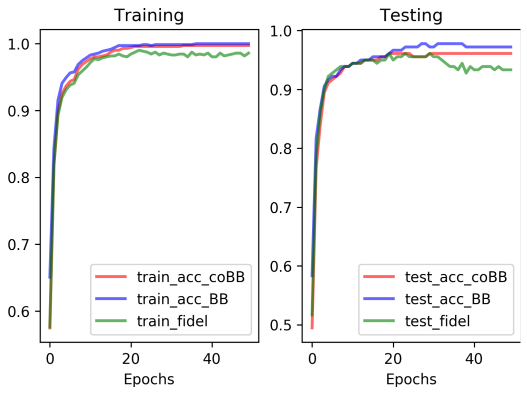
learning with STruGMA, our coBB models remain competitive in terms of predictive performance compared to decision trees.

6.3.4 Evolution of the Distance between the Black-box Model and STruGMA

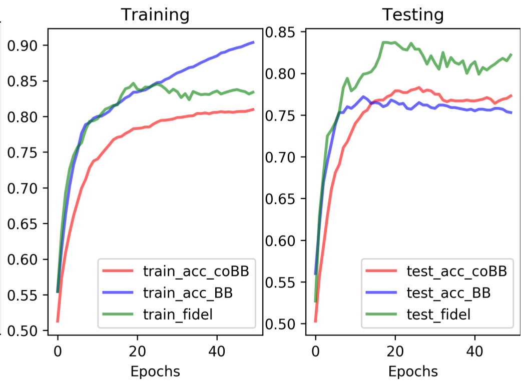
Figure 6.4 shows the evolution of the accuracy and fidelity over the first 50 co-learning iterations for each dataset. Despite the iterative nature of the co-learning that alternates between learning the black-box model and its STruGMA surrogate, the fidelity of the two models increases throughout iterations. This means that the main goal which is essentially to minimise the distance between the two models can be achieved with co-learning. Moreover, in Figure 6.5, the losses decrease properly and a local optimal can usually be reached after or even before 50 iterations of co-learning.

6.4 Conclusion and Future Work

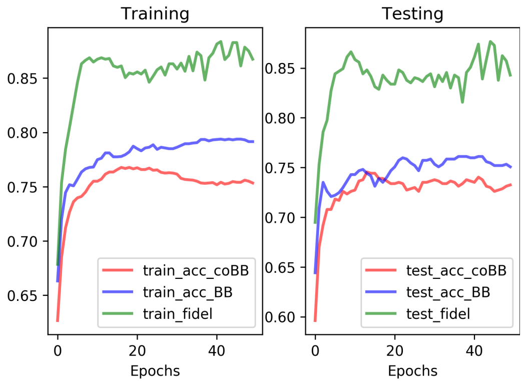
This paper chapter introduced a co-learning framework for global explanations of black-box models with decision rules. In this statistical framework, a black-box model is explained by co-learning a newly introduced soft truncated Gaussian mixture analysis (STruGMA) that encapsulates hyper-rectangle decision rules. Simultaneously, the black-box model is encouraged by a penalty term to satisfy the hyper-rectangle rules of STruGMA. Results show that our framework improves the fidelity of global explanations, while having a limited impact on the accuracy



(a) Wine



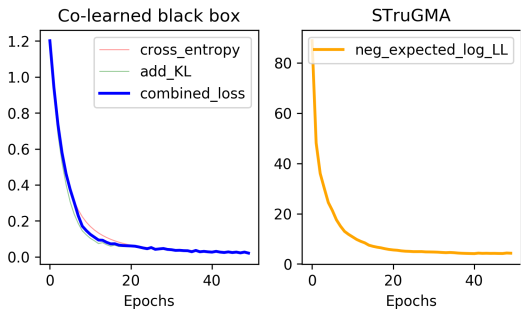
(b) Credit



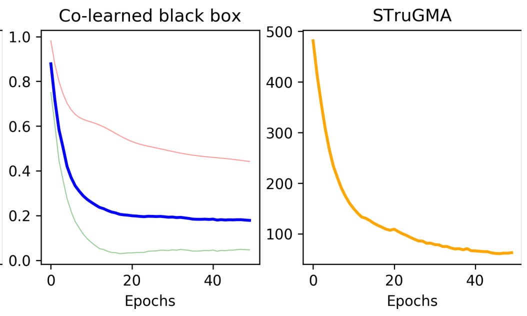
(c) Pima

Figure 6.4: Evolution of mean of accuracy and fidelity over five repetitions of the first 50 iterations of co-learning.

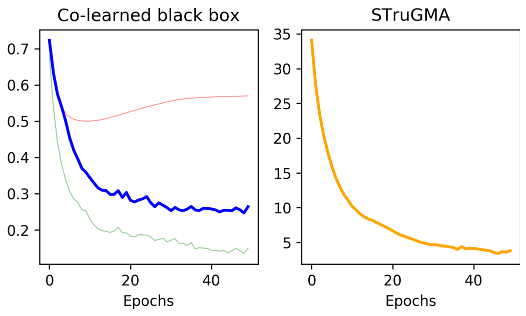
of the black-box model, which remains competitive. The framework also opens up a wide range of perspectives since it can be used for any black-box model trainable through gradient descent. Future works will consider other black-box models like SVMs and other rule learners to provide global explanations. In addition, one can directly inject strong priors on STruGMA to automatically get decision rules explanations. Finally, it will be interesting to perform experiments with image data to provide more faithful global explanations of deep CNNs in the light of Zhang *et al.* (2019).



(a) Wine



(b) Credit



(c) Pima

Figure 6.5: Evolution of the losses over the first 50 iterations of co-learning (one repetition/dataset). Cross_entropy is the cross-entropy of the co-learned black-box model whereas add_KL is the divergence between the same model and STruGMA. Combined_loss is the convex combination of cross_entropy loss and the add_KL loss whereas neg_expected_log_LL is the negative expected log-likelihood of the STruGMA. Plots (a–c) share the same legend.

Chapter 7

Discussion

This section discusses more deeply the findings of this thesis. We restate the known results from the literature, the new results revealed by our techniques and finally we discuss the limitations, which will serve us to provide future works in Section 8.

7.1 On the Use of Constraint Enforcement for Global Explainability

Motivated by the need for reliable global explainability tools to explain black-box models, we have introduced in Chapter 6, a co-learning framework where a differentiable black-box is *implicitely* regularised by its decision rules explanations.

Known results. The majority of previous studies for global explainability usually propose techniques to improve the faithfulness (e.g., fidelity) between their explanations and the black-box model. Few studies use known decision rules (provided by humans or inferred by rule models such as random forests) in order to further improve the faithfulness of explanations and the black-box model.

Novelty. Our approach presented in Chapter 6 does not assume any pre-existing rules to constrain the black-box model. This a first key improvement over state-of-the-art techniques. Our approach also reveals that when constraining the black-box model by its extracted rule explanations, the fidelity of the resulting explanations is improved while the accuracy of the black-box model is marginally impacted. Inspired by the multiple gradient descent algorithm (MGDA) (Sener and Koltun, 2018; Désidéri, 2009), our optimisation approach leads to a technique that does not assume any hyperparameter to control the degree of trade-off between the fidelity of explanations and the performance of the black-box model. Therefore, our technique has the interesting property of *auto-discovering* the trade-off.

Limitations. While offering improvements over state-of-the-art techniques, our technique has several limitations. The first limitation comes from our assumed choice of the type of black-box models that we assume to be differentiable. As a result, our method is not currently applicable to non-differentiale black-box models such as random forests. Another current limitation of our approach comes from the surrogate model called STruGMA. Indeed, in its current form, STruGMA is only suitable for tabular data. Therefore, in its current state, our approach will not probably work well with images and text data where unstructured input features lack a semantic meaning. Finally, our approach does not provide uncertainty of explanations, which may be useful in certain situations where we want probability estimates on how confident are explanation tools.

7.2 On the Imposition of Fairness Constraints On Top-down Greedy Decision Trees

Decision trees are widely used. Due of their non-smoothness, several machine learning techniques such as fairness-aware learning algorithms that assume differentiability are inapplicable to them. In Chapter 4, we introduce a boundary-based fairness decision tree learner to learn decision tree under the fairness constraint.

Known results. Several works have been proposed to impose fairness constraints on differentiable models. A notable work that we adapted is the work of Zafar *et al.* (2019), which can handle different fairness notions by measuring the degree of independence using covariance, and can also handle non-binary sensitive groups. Other works try to impose the fairness constraint on decision trees by designing fairness-aware heuristics such as the fair information gain (FIG) of Zhang and Ntoutsi (2019), which is effective in reducing the disparate impact or demographic parity unfairness.

Novelty. Our approach leverages the work of Zafar *et al.* (2019) to make it possible to learn decision trees under boundary-based fairness constraints. Thanks to the geometrical view of decision trees, it bypasses the assumption over the *explicit* expression of distance to decision boundary. Our new findings reveal that our method is more flexible by working better on different types of unfairness notions. Unlike to work of Zhang and Ntoutsi (2019), our technique makes it possible to play with an hyperparameter to achieve a trade-off between accuracy of classification trees and their unfairness (especially the disparate impact unfairness).

Limitations. The technique we presented has some drawbacks. The first one is a theoretically motivated limitation from the optimisation. Indeed, due to the use of a greedy optimisation with constraints, apart from experimental results, it is unknown whether our new penalised heuristic that we minimise is an upper bound on the *penalised misclassification loss*. For traditional heuristics such as Index Gini, these heuristics are upper bounds on misclassification loss and we have theoretical guarantees that, given a sufficient number of leaf nodes, optimising a local heuristic will eventually lead to a desired level of accuracy (Kearns and Mansour, 1999). For our proposed heuristic as well as previously proposed ones, this is an open problem. Another limitation of our approach directly comes from the limitation of Zafar *et al.* (2019). It is linked to the *proxy* evaluation of independence through Pearson correlation. However, correlation is not a measure of independence. Even if it may allow to assess a level of dependence, it is not able to provide answer over independence. Moreover, since we evaluate empirically the correlation, another limitation is the fact that we require several training examples to have low-variance estimate of the true correlation.

7.3 On the Integration of Domain-knowledge Constraints in Optimal Decision Tree Learners

Unlike top-down greedy decision tree learners that optimise a local heuristic in each internal node, optimal tree learners allow to learn decision trees using a global objective expressed over all leaf nodes. In Chapter 5, we propose a tree representation that leads to two generic formulations to learn optimal decision trees under a broad class of domain-knowledge constraints.

Known results. Several depth-constrained and size-constrained optimal tree learners have been recently proposed mostly to improve the computational time of the global optimisation. They introduce several design and technical mechanisms to reduce the optimisation time to find optimal decision trees. Examples are BinOCT (Verwer and Zhang, 2019), which uses a heuristic to discretise features, and DL8.5, which uses an item-set mining approach with branch-and-bound and caching.

Novelty. Instead of focusing on the computational time improvement, our framework is designed to ease the enforcement of a broad class of domain-knowledge constraints by also modeling the optimal decision tree learning problem with great flexibility. We show several use cases where we are able to enforce domain-knowledge constraints (fairness included) without (or with a very limited) loss in performance. Another interesting property of our tree representation is its ability to easily switch from binary and continuous split thresholds without drastic changes on the formulation.

Limitations. Since the optimisation of objective functions for decision trees is crucial, our natural first limitation comes from the optimisation tool. Indeed, having a MIP formulation to learn decision trees, our approach does not scale with large datasets and high number of features since the problem is NP-complete. In constraint-free settings, our framework also inherits from the high risk of overfitting caused by the oversearching of the optimal decision tree. This type of overfitting is orthogonal to the classical one whose source is related to model complexity (Quinlan and Cameron-Jones, 1995). Raised by Dietterich (1995) who claim that *"in machine learning, it is optimal to be sub-optimal"*, this issue has been theoretically investigated and discussed by Domingos (1999) and Jensen and Cohen (2000) who provide evidence that greater simplicity (i.e., lower complexity) does not necessarily lead to good generalisation. The learner or the learning algorithm has also its importance on the generalisation. Although it has been studied by the Quinlan (1986) in the context of rule list learning, the oversearching problem is severely understudied for decision tree learners. This may explain the evidence obtained in Chapter 3, which was the fact that top-down greedy methods are still competitive compared to constraint-free optimal decision tree learners with respect to predictive performance.

Chapter 8

Conclusion and Future Works

This chapter summarises the main contributions and results of the thesis guided by the research questions defined in Chapter 1. It also suggests future works derived from the limitations of our contributions highlighted in Chapter 7.

8.1 Summary of our Contributions and Results

This thesis focused on constraint enforcement on two specific machine learning models.

First, we considered decision tree models and imposed domain-knowledge constraints (fairness included). We made a thorough investigation and analysis of techniques available in the literature to answer the question how does the literature tackle the imposition of constraints on decision trees. We propose a taxonomy of constraints and another taxonomy of approaches through the lens of optimisation tools employed by these methods to learn decision trees. The taxonomy of constraints defined structure-level constraints (e.g., size, depth), feature-level constraints (e.g., monotonicity, fairness, privacy), instance-level constraints (e.g., robustness). The taxonomy of methods include top-down greedy, safe enumeration, LP/SAT/CP and probabilistic (including Bayesian) approaches. While the top-down greedy approach has been heavily leveraged to learn constrained decision trees, we found that the literature is scared on using global tree learners with LP/SAT/CP despite they have they great advantage, when modelled, to easily integrate constraints. We also observe several research directions that should be done to answer open questions. Finally, we zoomed onto depth-constrained decision tree learners and made experiments to benchmark their predictive performance. Experiments show that top-down greedy methods such as CART are still competitive with respect to recently proposed optimal decision tree learners w.r.t. predictive performance.

Second, still considering decision tree models, we propose two techniques to enforce constraints on decision trees. The first technique called BDT specifically introduces a boundary-based fairness constraints to learn fairer decision trees. It is a soft constraint enforcement and uses a top-down greedy approach to learn decision trees. Results show that our approach is competitive with respect to a state-of-the art method, while having the advantage of being much more flexible (working better with two fairness notions). The second technique called CPTree introduces a framework based on MIP/CP to learn decision trees under domain-knowledge constraints. This one is a hard constraint enforcement and results show that in constraint-free settings our framework learns CPTrees that are competitive with respect to state-of-the-art trees and by theoretically formalising a broad class of constraints, we show that our framework is flexible enough to easily integrate domain-knowledge constraints. We also provide experimental evidence through use cases that when enforcing constraints CPtrees remain competitive in terms of predictive performance.

Third, considering differentiable black-box models such as MLPs, we introduced a statistical framework where these models can be implicitly constrained to be easily explainable by decision rules. This is a soft constraint enforcement and the framework is fully differentiable. Compared to standard *posthoc* explainability setting, we provide experimental evidence that our framework usually achieve best results in terms of faithfulness (evaluated with fidelity) of explanations, while having a limited impact on the constrained black-box model compared to the unconstrained one.

8.2 Short-term Future Work

In the following, we provide short-term future works that should be done in order to complete the work done in this thesis.

Library or Software to Ease Modern Decision Tree Fitting. Decision trees and their ensemble models are widely used in real-world applications with and without constraint enforcement. However, as shown in the Table 2 of our survey (**Nanfack** *et al.*, 2022a), their implementations are scattered into several programming languages, several frameworks. We also built our own frameworks from scratch with little to no matching with respect to previous ones. This is because different frameworks (e.g., Scikit-learn (Buitinck *et al.*, 2013), DL8.5 (Aglin *et al.*, 2020), etc.) use completely different and highly optimised C libraries to speed up the optimisation. Having a unified and easily extendable framework will not only benefit a lot to the research community but it will deeply increase the applicability of tree models in several domains.

Constraint Inference. One major limitation of the real-world application of our constrained CPTrees is that they rely on prior knowledge that should be *in phase* with training data. Otherwise, constraints will likely impair performance of the model. However, acquiring this domain knowledge is a tedious task. One extension of our framework is to reformulate the framework such that we automatically infer constraint from data. Even though it violates the Bayesian principle which says that prior knowledge should not come from data, it has the potential to not relying on domain experts to provide such knowledge but instead, it should, at least, give propositions for validation by domain experts. Furthermore, for user-centric applications, it is much more easier to grasp knowledge in terms of constraints.

Extension to Random Forests. Although, we principally focused on the decision tree hypothesis class for techniques presented in Chapter 4 and Chapter 5 (namely BDT and CPTree), their extension to random forests models is not difficult, especially for BDT. In the case of CPTree, one should pay attention to the computational time and therefore, adapt the sub-sampling step for random forests taking into account this computational time.

Accelerating Computations for End-user Interaction and Validation. Implementation of the techniques presented in Chapter 4 and Chapter 5 (namely BDT and CPTree) are far to be at their optimal computational time and complexity. This computational issue may reduce their impact when practitioners may want to use them in the scenario of interactive machine learning. Additionally, while in Chapter 5, we mostly acquired prior knowledge from a medical expertise, in a more general setting, it is suitable to directly involve the end-user expertise for validation of learned trees. Here the advantage of using decision trees is that their interpretability makes them suitable to easily formalise a sort of *human-in-the-loop* prior (Lage *et al.*, 2018) to model, e.g., label correction in an explainable way (Esmeir and Markovitch, 2022).

Uncertainty of Explanations. The co-learning framework we proposed in Chapter 6 is in the pure frequentist logic. It does not model uncertainty of explanations, which is a key ingredient for trustworthy explanations. Further studies should be done to enable uncertainty of explanations, through for example the use of Bayesian rule learners.

Co-learning and Adapting STruGMA For Image data. In order to integrate the explainability constraint in the through the co-learning framework, we introduced STruGMA, which is a differentiable model that aim to encapsulate decision rules. In Chapter 7, we highlighted, that in its current form, STruGMA will probably not provide good results with image data. Future studies should be done to "*deepify*" (make STruGMA compatible to deep neural network modules) STruGMA such that it would be able to handle image data. For text data, a similar co-learning called Unirex (Chan *et al.*, 2022) (composed with a similar module to STruGMA) has been recently released by the Meta (ex Facebook) company. However, to our knowledge, this kind of framework does not exist for image data yet.

8.3 Long-term Future Work

This sections presents long-term future works that should be done in order to increase the impact of the contributions made in this thesis.

Theoretical Guarantees of Constraint Enforcement. In this thesis, readers observed that most of our guarantees in terms of the predictive performance and to some extent satisfaction of constraints were empirical. For some reasons that include the number of features, the number of training examples, we may observe different behaviours when enforcing constraints. Indeed, the classical machine learning theory usually only provide guarantees over the learnability in constraint-free settings (Ben-David, 1995). Due to nowadays wide applicability of machine learning, there is a very recent interest Chamon and Ribeiro (2020); Chamon *et al.* (2022) in trying to fill the gap by providing extensions of the learnability theory under constraints that include fairness. However, there is no work for example that studies this question for decision tree models.

Sensitive Analysis of Constraint Enforcement. Not only in this thesis, but also in the machine learning community, it is usually common to enforce several constraints in isolation i.e., one at a time. Examples include the works of Cotter *et al.* (2019) and Yang *et al.* (2020) and ours. As shown by the *impossibility* *theorem* for fairness in machine learning, there exists couples of constraints that are pairwise incompatible. Worse, we hypothesise that when pairing constraints they may be even harmful for the learning task. To what extents mixing different types of constraints (e.g., fairness, explanaibility, complexity, privacy, etc.) may harm the learning task? This is a wide open question and research needs to be done in order to truly assess the sensibility and the importance of constraint enforcement.

Rethinking the Generalisation of Decision Trees. The classical theory of generalisation of classical machine learning models is usually provided without considering the optimisation tool used to fit models. However, one finding of this thesis, revealed by benchmarking optimal tree learners and by visualising the optimisation paths, was the high risk of overfitting from optimal tree learners. As a result, the optimisation tool has a great importance on the generalisation of machine learning models. For example, on deep neural networks models, it is hypothesised and demonstrated empirically that stochastic gradient descent is an implicit regulariser (Ali *et al.*, 2020). We also hypothesise a similar statement for top-down greedy methods such as CART. However, further studies should be done to theoretically analyse the generalisation optimal tree learners and propose mechanisms to counter the overfitting from oversearching through e.g., explicit constraints.

On the Path to Explicit Explainability Constraint Terms. In machine learning, the two most classical and principled ways to infuse prior knowledge in models are to formalise either the prior distribution $p(\boldsymbol{\theta})$ or a constraint function $g(\boldsymbol{\theta})$ over the model parameters $\boldsymbol{\theta}$. However, formalising this constraint function or this prior distribution is far to be straightforward. That is why, constraints are often enforced *implicitly* as in Chapter 6. For interpretable models such as decision trees or linear models, the explainability constraints is often handled through the complexity (e.g., number of leaves or sparsity of weights). For complex black-box models such as random forests or neural networks, it is relatively unknown whether such statement is valid. Moreover, the issue of how to formalise the learning problem under an explicit constraint, e.g., "that the hypothesis is easily explainable or interpretable" is still an open problem, although the work of Dziugaite *et al.* (2020) has tried to provide ideas in this direction.

Bibliography

Aghaei, S., Azizi, M. J., and Vayanos, P. (2019). Learning optimal and fair decision trees for non-discriminative decision-making. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1418–1426, Honolulu, Hawaii, USA. AAAI Press. 38, 52, 53, 57, 59, 64, 65, 66, 76, 89, 91

Aghaei, S., Gómez, A., and Vayanos, P. (2021). Strong optimal classification trees. *arXiv preprint arXiv:2103.15965*. 57, 62, 63, 64

Aglin, G., Nijssen, S., and Schaus, P. (2020). Learning optimal decision trees using caching branch-and-bound search. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3146–3153. AAAI Press. 44, 62, 63, 90, 113, 118, 147

Aïvodji, U., Arai, H., Fortineau, O., Gambs, S., Hara, S., and Tapp, A. (2019). Fairwashing: the risk of rationalization. In *International Conference on Machine Learning*, pages 161–170. PMLR. 40

Ali, A., Dobriban, E., and Tibshirani, R. (2020). The implicit regularization of stochastic gradient flow for least squares. In *International conference on machine learning*, pages 233–244. PMLR. 149

Alvarez, I., Bernard, S., and Deffuant, G. (2007). Keep the decision tree and estimate the class probabilities using its decision boundary. In *Proc. of IJCAI*, pages 654–659. 78

Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., and Rudin, C. (2018). Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, **18**(234), 1–78. 45, 90

Angelopoulos, N. and Cussens, J. (2005a). Exploiting informative priors for bayesian classification and regression trees. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 641–646, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 46, 57, 68

Angelopoulos, N. and Cussens, J. (2005b). Tempering for bayesian c&rt. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 17–24, New York, NY, USA. ACM Press. 46, 57

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016). Machine bias. In *Ethics of Data and Analytics*, pages 254–264. Auerbach Publications. 81

Arbatli, A. D. and Akin, H. L. (1997). Rule extraction from trained neural networks using genetic algorithms. *Nonlinear Analysis: Theory, Methods & Applications*, **30**(3), 1639–1648. 125

Auer, P., Holte, R. C., and Maass, W. (1995). Theory and applications of agnostic pac-learning with small decision trees. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, ICML'95, pages 21–29, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 44, 57, 59, 65

Avellaneda, F. (2020). Efficient inference of optimal decision trees. In *34th AAAI Conference on Artificial Intelligence*. AAAI Press. 45, 57, 64, 91

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, **58**, 82 – 115. 38, 89, 92

Barros, R. C., Basgalupp, M. P., de Carvalho, A. C. P. L. F., and Freitas, A. A. (2012). A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, **42**(3), 291–312. 40

Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A. V., and Criminisi, A. (2016). Measuring neural net robustness with constraints. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2621–2629, USA. Curran Associates Inc. 66

Ben-David, A. (1995). Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, **19**(1), 29–43. 48, 56, 148

Bennett, K. P. and Blue, J. A. (1996). Optimal decision trees. Technical report, R.P.I. Math Report No. 214, Rensselaer Polytechnic Institute. 43, 57, 59

Bennett, K. P. and Mangasarian, O. L. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, **1**(1), 23–34. 57

Bertsimas, D. and Dunn, J. (2017). Optimal classification trees. *Machine Learning*, **106**(7), 1039–1082. 44, 45, 47, 57, 59, 62, 64, 65, 89, 91, 99, 112

Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA. 21, 27, 29

Bessiere, C., Hebrard, E., and O'Sullivan, B. (2009). Minimising decision tree size as combinatorial optimisation. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, CP'09, pages 173–187, Berlin, Heidelberg. Springer-Verlag. 37, 43, 57, 59, 64

Biggio, B. and Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, **84**, 317–331. 66

Blockeel, H., Raedt, L. D., and Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 55–63, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 12

Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press. 21

Boz, O. (2002). Extracting decision trees from trained neural networks. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 456–461, New York, NY, USA. ACM Press. 56, 58, 66, 125

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA. 13, 62, 63, 90, 114

Buhrman, H. and De Wolf, R. (2002). Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, **288**(1), 21–43. 40

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122. 147

Buntine, W. (1992). Learning classification trees. *Statistics and computing*, **2**(2), 63–73. 57, 60, 62

Calders, T., Kamiran, F., and Pechenizkiy, M. (2009). Building classifiers with independency constraints. In *Proc. of ICDM Workshop on Domain Driven Data Mining*, pages 13–18. 76

Calmon, F. P., Wei, D., Vinzamuri, B., Ramamurthy, K. N., and Varshney, K. R. (2017). Optimized pre-processing for discrimination prevention. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3995–4004. 75

Calzavara, S., Lucchese, C., and Tolomei, G. (2019). Adversarial training of gradient-boosted decision trees. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2429–2432. 54, 56

Calzavara, S., Lucchese, C., Tolomei, G., Abebe, S. A., and Orlando, S. (2020). Treant: training evasion-aware decision trees. *Data Mining and Knowledge Discovery*, **34**(5), 1390–1420. 54, 56

Cardoso, J. S. and Sousa, R. (2010). Classification models with global constraints for ordinal data. In *Proceedings of the 2010 Ninth International Conference on Machine Learning and Applications*, ICMLA '10, pages 71–77, Washington, DC, USA. IEEE Computer Society. 48, 56

Chai, J. and Wang, X. (2022). Fairness with adaptive weights. In *International Conference on Machine Learning*, pages 2853–2866. PMLR. 75

Chamon, L. and Ribeiro, A. (2020). Probably approximately correct constrained learning. *Advances in Neural Information Processing Systems*, **33**, 16722–16735. 148

Chamon, L. F. O., Paternain, S., Calvo-Fullana, M., and Ribeiro, A. (2022). Constrained learning with non-convex losses. *IEEE Transactions on Information Theory*, pages 1–1. 148

Chan, A., Sanjabi, M., Mathias, L., Tan, L., Nie, S., Peng, X., Ren, X., and Firooz, H. (2022). Unirex: A unified learning framework for language model rationale extraction. In *International Conference on Machine Learning*, pages 2867–2889. PMLR. 148

Chen, H., Zhang, H., Boning, D., and Hsieh, C.-J. (2019). Robust decision trees against adversarial examples. In *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1122–1131, Long Beach, California, USA. PMLR. 54, 56, 66

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*. 124

Cheng, M., Le, T., Chen, P.-Y., Zhang, H., Yi, J., and Hsieh, C.-J. (2019). Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representation (ICLR)*, New Orleans, LA, USA. 54

Chipman, H. A., George, E. I., and McCulloch, R. E. (1998). Bayesian cart model search. *J. Amer. Statist. Assoc.*, **93**(443), 935–948. 46, 57, 60, 62

Choi, E., Bahadori, M. T., Kulas, J. A., Schuetz, A., Stewart, W. F., and Sun, J. (2016). Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 3512–3520, USA. Curran Associates Inc. 58

Chouldechova, A. and Roth, A. (2020). A snapshot of the frontiers of fairness in machine learning. *Communications of the ACM*, **63**(5), 82–89. 51, 52, 74

Cohen Jr, A. C. (1950). Estimating the mean and variance of normal populations from singly truncated and doubly truncated samples. *The Annals of Mathematical Statistics*, pages 557–569. 128

Confalonieri, R., Weyde, T., Besold, T. R., and del Prado Martín, F. M. (2020). Trepan reloaded: A knowledge-driven approach to explaining artificial neural networks. In *24th European Conference on Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2457–2464. IOS Press. 125

Conti, M., Di Pietro, R., Mancini, L. V., and Mei, A. (2009). (new) distributed data source verification in wireless sensor networks. *Inf. Fusion*, **10**(4), 342–353. 82

Cotter, A., Jiang, H., Gupta, M., Wang, S., Narayan, T., You, S., and Sridharan, K. (2019). Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research*, **20**(172), 1–59. 39, 104, 105, 112, 148

Cplex, I. I. (2009). V12. 1: User's manual for cplex. *International Business Machines Corporation*, **46**(53), 157. 29

Craven, M. and Shavlik, J. W. (1994). Using sampling and queries to extract rules from trained neural networks. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, page 37–45, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 125

Craven, M. W. and Shavlik, J. W. (1995). Extracting tree-structured representations of trained networks. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, pages 24–30, Cambridge, MA, USA. MIT Press. 43, 56, 58, 66, 125

Craven, M. W. and Shavlik, J. W. (1996). *Extracting Comprehensible Models from Trained Neural Networks*. Ph.D. thesis, The University of Wisconsin - Madison. AAI9700774. 125

Daniëls, H. and Velikova, M. (2003). Derivation of monotone decision models from non-monotone data. Technical report, Tilburg University, Center for Economic Research. 56

Das, G. and Goodrich, M. T. (1997). On the complexity of optimization problems for 3-dimensional convex polyhedra and decision trees. *Computational Geometry*, **8**(3), 123–137. 12

Davis, J. V., Ha, J., Rossbach, C. J., Ramadan, H. E., and Witchel, E. (2006). Cost-sensitive decision tree learning for forensic classification. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 622–629, Berlin, Heidelberg. Springer-Verlag. 49, 56

Delobelle, P., Temple, P., Perrouin, G., Frénay, B., Heymans, P., and Berendt, B. (2021). Ethical adversaries: Towards mitigating unfairness with adversarial machine learning. *SIGKDD Explor. Newsl.*, **23**(1), 32–41. 84

Denison, D. G., Mallick, B. K., and Smith, A. F. (1998). A bayesian cart algorithm. *Biometrika*, **85**(2), 363–377. 57, 60, 62

Désidéri, J.-A. (2009). *Multiple-Gradient Descent Algorithm (MGDA)*. Ph.D. thesis, INRIA. 132, 141

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. 124

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, **27**(3), 326–327. 144

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, **40**(2), 139–157. 14

Domingos, P. (1999). The role of occam's razor in knowledge discovery. *Data mining and knowledge discovery*, **3**(4), 409–425. 144

Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J., and Pontil, M. (2018). Empirical risk minimization under fairness constraints. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2796–2806. 75

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv*. 124

Du, W. and Zhan, Z. (2002). Building decision tree classifier on private data. In *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining - Volume 14*, CRPIT '14, pages 1–8, Darlinghurst, Australia, Australia. Australian Computer Society, Inc. 50, 81

Dua, D. and Graff, C. (2017). UCI machine learning repository. 38, 63, 105, 106, 108, 112, 133

Dziugaite, G. K., Ben-David, S., and Roy, D. M. (2020). Enforcing interpretability and its statistical impacts: Trade-offs between accuracy and interpretability. *arXiv preprint arXiv:2010.13764*, **abs/2010.13764**. 38, 89, 149

Esmeir, S. and Markovitch, S. (2004). Lookahead-based algorithms for anytime induction of decision trees. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 33–, New York, NY, USA. ACM Press. 49, 57, 59

Esmeir, S. and Markovitch, S. (2006). Any time induction of decision trees: An iterative improvement approach. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 348–355, Boston, Massachusetts, USA. AAAI Press. 49, 57, 59

Esmeir, S. and Markovitch, S. (2022). Explainable and local correction of classification models using decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI'22. AAAI Press. 147

Estruch, V., Ferri, C., Hernández-Orallo, J., and Ramirez-Quintana, M. (2002). Re-designing cost-sensitive decision tree learning. In *Workshop de Mineria de Datos y Aprendizaje*, pages 33–42. 49, 57, 59

Feelders, A. and Pardoel, M. (2003). Pruning for monotone classification trees. In *5th International Symposium on Intelligent Data Analysis*, IDA '03, pages 1–12, Berlin, Heidelberg. Springer-Verlag. 48, 56

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., and Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268. 72

Ferri, C., Flach, P. A., and Hernández-Orallo, J. (2002). Learning decision trees using the area under the roc curve. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 139–146, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 49, 56

Firat, M., Crognier, G., Gabor, A., Hurkens, C., and Zhang, Y. (2020). Column generation based heuristic for learning classification trees. *Computers & Operations Research*, **116**. 57, 59

Fletcher, S. and Islam, M. Z. (2019). Decision tree classification with differential privacy: A survey. *ACM Comput. Surv.*, **52**(4), 83:1–83:33. 53

Floridi, L. (2019). Establishing the rules for building trustworthy ai. *Nature Machine Intelligence*, **1**(6), 261–262. 89

Frank, E. and Witten, I. H. (1999). Making better use of global discretization. In *16th International Conference on Machine Learning (ICML 99)*, pages 115–123. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 119

Freitas, A., Costa-Pereira, A., and Brazdil, P. (2007). Cost-sensitive decision trees applied to medical data. In *Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery*, DaWaK'07, pages 303–312, Berlin, Heidelberg. Springer-Verlag. 49, 56

Freitas, A. A. (2014). Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, **15**(1), 1–10. 36, 47, 48, 65, 66, 92

Friedman, A. and Schuster, A. (2010). Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 493–502, New York, NY, USA. ACM Press. 51, 53

Friedman, A., Schuster, A., and Wolff, R. (2006). K-anonymous decision tree induction. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, ECMLPKDD'06, pages 151–162, Berlin, Heidelberg. Springer-Verlag. 50, 56

Fromont, E., Blockeel, H., and Struyf, J. (2007). Integrating decision tree learning into inductive databases. In *Proceedings of the 5th International Conference on Knowledge Discovery in Inductive Databases*, KDID'06, pages 81–96, Berlin, Heidelberg. Springer-Verlag. 41, 43, 49, 57, 59

Fürnkranz, J., Gamberger, D., and Lavrač, N. (2012). *Foundations of rule learning*. Springer Science & Business Media. 128

Gangrade, A. and Patel, R. (2009). Building privacy-preserving c4. 5 decision tree classifier on multi-parties. *International Journal on Computer Science and Engineering*, **1**(3), 199–205. 51, 56

Gangrade, A. and Patel, R. (2012). Privacy preserving two-layer decision tree classifier for multiparty databases. *International Journal of Computer and Information Technology (2277-0764)*, **1**(1), 77–82. 50, 56

Garofalakis, M., Hyun, D., Rastogi, R., and Shim, K. (2000). Efficient algorithms for constructing decision trees with constraints. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 335–339, New York, NY, USA. ACM Press. 41, 43, 56, 58, 62

Garofalakis, M., Hyun, D., Rastogi, R., and Shim, K. (2003). Building decision trees with constraints. *Data Min. Knowl. Discov.*, **7**(2), 187–214. 41, 43, 57, 58, 59

Gehrke, J., Ganti, V., Ramakrishnan, R., and Loh, W.-Y. (1999). Boa–optimistic decision tree construction. *SIGMOD Rec.*, **28**(2), 169–180. 56, 57, 59

Geurts, P. and Wehenkel, L. (2000). Investigation and reduction of discretization variance in decision tree induction. In *European Conference on Machine Learning*, pages 162–170. Springer. 119

Gopinath, D., Converse, H., Pasareanu, C., and Taly, A. (2019). Property inference for deep neural networks. In *2019 34th IEEE/ACul International Conference on Automated Software Engineering (ASE)*. 125

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A survey of methods for explaining black box models. *Comput. Surveys*, **51**(5), 93:1–93:42. 39, 65, 66, 124

Hardt, M., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. *Advances in neural information processing systems*, **29**. 52, 75

Hastie, T. and Tibshirani, R. (1996). Discriminant analysis by gaussian mixtures. *Journal of the Royal Statistical Society: Series B (Methodological)*, **58**(1), 155–176. 128

Heidenberger, K. (1996). Dynamic project selection and funding under risk: A decision tree based milp approach. *European Journal of Operational Research*, **95**(2), 284–298. 57, 59, 62

Hodgson, J. M., van Someren, V. H., Smith, C., and Goyale, A. (2018). Direct bilirubin levels observed in prolonged neonatal jaundice: a retrospective cohort study. *BMJ paediatrics open*, **2**(1). 101

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, **4**(2), 251–257. 15

Hu, H., Siala, M., Hébrard, E., and Huguet, M.-J. (2020). Learning optimal decision trees with maxsat and its integration in adaboost. In *IJCAI-PRICAI 2020, 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence*. 45, 57, 62, 63

Hu, Q., Che, X., Zhang, L., Zhang, D., Guo, M., and Yu, D. (2012). Rank entropy-based decision trees for monotonic classification. *IEEE Transactions on Knowledge and Data Engineering*, **24**(11), 2052–2064. 48, 53, 61

Hu, Q. H., Guo, M. Z., Yu, D. R., and Liu, J. F. (2010). Information entropy for ordinal classification. *Science in China, Series F: Information Sciences*, **53**(6), 1188–1200. 53, 61

Hu, X., Rudin, C., and Seltzer, M. (2019). Optimal sparse decision trees. In *Advances in Neural Information Processing Systems 32*, pages 7265–7273. Curran Associates, Inc. 45, 57, 62, 63, 90, 91, 113, 114, 118

Hu, Z., Yang, Z., Salakhutdinov, R., and Xing, E. (2016). Deep neural networks with massive learned knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1670–1679. 126

Hyafil, L. and Rivest, R. L. (1976). Constructing optimal binary decision trees is np-complete. *Information processing letters*, **5**(1), 15–17. 12

Iqbal, M. R. A., Rahaman, M. S., and Nabil, S. I. (2012). Construction of decision trees by using feature importance value for improved learning performance. In *Proceedings of the 19th International Conference on Neural Information Processing - Volume Part II*, ICONIP'12, pages 242–249, Berlin, Heidelberg. Springer-Verlag. 56

Jeff, L., Surya, M., Lauren, K., and Julia, A. (2016). How we analyzed the compas recidivism algorithm. 81

Jensen, D. D. and Cohen, P. R. (2000). Multiple comparisons in induction algorithms. *Machine Learning*, **38**(3), 309–338. 144

Kallus, N. and Zhou, A. (2018). Residual unfairness in fair machine learning from prejudiced data. In *International Conference on Machine Learning*, pages 2439–2448. PMLR. 81

Kamiran, F., Calders, T., and Pechenizkiy, M. (2010). Discrimination aware decision tree learning. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 869–874, Washington, DC, USA. IEEE Computer Society. 38, 52, 53, 56, 58, 61, 76

Kamp, R., Feelders, A., and Barile, N. (2009). Isotonic classification trees. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, IDA '09, pages 405–416, Berlin, Heidelberg. Springer-Verlag. 48, 56

Kantchelian, A., Tygar, J. D., and Joseph, A. D. (2016). Evasion and hardening of tree ensemble classifiers. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2387–2396, New York, NY, USA. JMLR Press. 54, 66

Kearns, M. and Mansour, Y. (1999). On the boosting ability of top–down decision tree learning algorithms. *Journal of Computer and System Sciences*, **58**(1), 109–128. 85, 143

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., *et al.* (2018). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pages 2668–2677. 125

Kim, M. P., Ghorbani, A., and Zou, J. (2019). Multiaccuracy: Black-box post-processing for fairness in classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 247–254. 75

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, San Diego, CA, USA. 32, 134

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 972–981, Red Hook, NY, USA. 124

Kleinberg, J., Mullainathan, S., and Raghavan, M. (2016). Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*. 76

Kocev, D., Struyf, J., and Džeroski, S. (2007). Beam search induction and similarity constraints for predictive clustering trees. In *Proceedings of the 5th International Conference on Knowledge Discovery in Inductive Databases*, KDID'06, pages 134–151, Berlin, Heidelberg. Springer-Verlag. 57, 59

Kononenko, I., Bratko, I., and Kukar, M. (1997). Application of machine learning to medical diagnosis. *Machine learning and data mining: Methods and applications*, **389**, 408. 89

Krętowski, M. and Grześ, M. (2006). Evolutionary induction of cost-sensitive decision trees. In *Proceedings of the 16th International Conference on Foundations of Intelligent Systems*, ISMIS'06, pages 121–126, Berlin, Heidelberg. Springer-Verlag. 49, 57, 59

Kruegel, C. and Toth, T. (2003). Using decision trees to improve signature-based intrusion detection. In *6th International Symposium on Recent Advances in Intrusion Detection*, RAID '03, pages 173–191, Berlin, Heidelberg. Springer-Verlag. 50, 56

Lage, I., Ross, A. S., Kim, B., Gershman, S. J., and Doshi-Velez, F. (2018). Human-in-the-loop interpretability prior. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Neurips'18, page 10180–10189. Curran Associates Inc. 147

Lahoti, P., Beutel, A., Chen, J., Lee, K., Prost, F., Thain, N., Wang, X., and Chi, E. H. (2020). Fairness without demographics through adversarially reweighted learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 728–740. 75

Lakkaraju, H., Bach, S. H., and Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1675–1684, New York, NY, USA. Association for Computing Machinery. 128

Li, Q., Wen, Z., and He, B. (2020a). Practical federated gradient boosting decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4642–4649. 50, 56

Li, Q., Wu, Z., Wen, Z., and He, B. (2020b). Privacy-preserving gradient boosting decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 784–791. 51, 56

Li, R., Daniel, R., and Rachet, B. (2016). How much do tumor stage and treatment explain socioeconomic inequalities in breast cancer survival? applying causal mediation analysis to population-based data. *European journal of epidemiology*, **31**(6), 603–611. 105

Li, X., Zhao, H., and Zhu, W. (2015). A cost sensitive decision tree algorithm with two adaptive mechanisms. *Knowledge-Based Systems*, **88**, 24–33. 49, 56

Lin, J., Zhong, C., Hu, D., Rudin, C., and Seltzer, M. (2020). Generalized and scalable optimal sparse decision trees. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6150–6160. PMLR. 57, 91

Ling, C. X., Yang, Q., Wang, J., and Zhang, S. (2004). Decision trees with minimal costs. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 69–77, New York, NY, USA. ACM Press. 49, 56

Liu, H., Hussain, F., Tan, C. L., and Dash, M. (2002). Discretization: An enabling technique. *Data mining and knowledge discovery*, **6**(4), 393–423. 92

Liu, L., Kantarcioglu, M., and Thuraisingham, B. (2009). Privacy preserving decision tree mining from perturbed data. In *2009 42nd Hawaii International Conference on System Sciences*, pages 1–10, Big Island, HI, USA. IEEE. 38, 51, 56

Lohaus, M., Perrot, M., and Von Luxburg, U. (2020). Too relaxed to be fair. In *37th International Conference on Machine Learning*. 104, 105

Lomax, S. and Vadera, S. (2013). A survey of cost-sensitive decision tree induction algorithms. *Comput. Surveys*, **45**(2), 16:1–16:35. 40, 49

López-Vallverdú, J. A., RiañO, D., and Collado, A. (2007). Increasing acceptability of decision trees with domain attributes partial orders. In *Proceedings of the Twentieth IEEE International Symposium on Computer-Based Medical Systems*, CBMS '07, pages 569–574, Washington, DC, USA. IEEE Computer Society. 39, 47, 50, 56, 66, 89

López-Vallverdú, J. A., RiañO, D., and Bohada, J. A. (2012). Improving medical decision trees by combining relevant health-care criteria. *Expert Syst. Appl.*, **39**(14), 11782–11791. 39, 41, 56, 66, 103

Luenberger, D. G., Ye, Y., *et al.* (1984). *Linear and nonlinear programming*, volume 2. Springer. 21, 22, 23

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2020). From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, **2**(1), 56–67. 124

Madzarov, G., Gjorgjevikj, D., and Chorbev, I. (2009). A multi-class svm classifier utilizing binary decision tree. *Informatica*, **33**(2), 233–241. 50, 57, 59

Marsala, C. and Petturiti, D. (2015). Rank discrimination measures for enforcing monotonicity in decision tree induction. *Inf. Sci.*, **291**(C), 143–171. 48, 53

Matwin, S., Felty, A., Hernádvölgyi, I., and Capretta, V. (2005). Privacy in data mining using formal methods. In *Proceedings of the 7th International Conference on Typed Lambda Calculi and Applications*, TLCA'05, pages 278–292, Berlin, Heidelberg. Springer-Verlag. 51, 56

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, **54**(6), 1–35. 5

Melis, D. A. and Jaakkola, T. (2018). Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, pages 7775–7784. 125

Menickelly, M., Günlük, O., Kalagnanam, J., and Scheinberg, K. (2016). Optimal generalized decision trees via integer programming. *CoRR*. 45, 57, 59

Mita, G., Papotti, P., Filippone, M., and Michiardi, P. (2020). LIBRE: Learning interpretable boolean rule ensembles. In *23rd International Conference on Artificial Intelligence and Statistics, 3-5 June 2020, Palermo, Sicily, Italy*, Palermo, ITALY. 124

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press. 9, 12, 16, 17

Murphy, K. P. (2022). *Probabilistic machine learning: an introduction*. MIT press. 12

Narodytska, N., Ignatiev, A., Pereira, F., and Marques-Silva, J. (2018). Learning optimal decision trees with sat. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 1362–1368. AAAI Press. 43, 44, 45, 57, 59, 64, 89, 91

Nijssen, S. (2008). Bayes optimal classification for decision trees. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 696–703, New York, NY, USA. ACM Press. 46, 57, 60, 68

Nijssen, S. and Fromont, E. (2007). Mining optimal decision trees from itemset lattices. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 530–539, New York, NY, USA. ACM Press. 43, 45, 46, 49, 57, 59, 90

Nijssen, S. and Fromont, E. (2010). Optimal constraint-based decision tree induction from itemset lattices. *Data Mining and Knowledge Discovery*, **21**(1), 9–51. 41, 43, 57, 90

Norouzi, M., Collins, M. D., Johnson, M., Fleet, D. J., and Kohli, P. (2015). Efficient non-greedy optimization of decision trees. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1729–1737, Cambridge, MA, USA. MIT Press. 50, 60

Norton, S. W. (1989). Generating better decision trees. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'89, pages 800–805, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 49, 56

Núñez, M. (1991). The use of background knowledge in decision tree induction. *Mach. Learn.*, **6**(3), 231–250. 41, 49, 50, 56, 62, 66, 89, 102

Nuti, G., Rugama, L. A. J., and Cross, A.-I. (2019). Efficient bayesian decision tree algorithm. *Corr*. 57, 60

Okajima, Y. and Sadamasa, K. (2019). Deep neural networks constrained by decision rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2496–2505. 125

Okunrintemi, V., Valero-Elizondo, J., Patrick, B., Salami, J., Tibuakuu, M., Ahmad, S., Ogunmoroti, O., Mahajan, S., Khan, S. U., Gulati, M., *et al.* (2018). Gender differences in patient-reported outcomes among adults with atherosclerotic cardiovascular disease. *Journal of the American Heart Association*, **7**(24), e010498. 109

Omielan, A. and Vadera, S. (2012). Ecco: A new evolutionary classifier with cost optimisation. In *7th IFIP TC 12 International Conference*, IIP '12, pages 97–105, Berlin, Heidelberg. Springer-Verlag. 49, 57, 59

Optimization, G. (2021). Gurobi optimizer reference manual. `http://www.gurobi.com`. 28, 114

Papernot, N., McDaniel, P. D., and Goodfellow, I. J. (2016). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*. 54, 66

Pasquale, F. (2015). *The Black Box Society: The Secret Algorithms That Control Money and Information*. Harvard University Press, Cambridge, MA, USA. 66

Pazzani, M. J., Merz, C. J., Murphy, P. M., Ali, K. M., Hume, T., and Brunk, C. (1994). Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, pages 217–225, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 49, 56

Pedapati, T., Balakrishnan, A., Shanmugam, K., and Dhurandhar, A. (2020). Learning global transparent models consistent with local contrastive explanations. In *Advances in Neural Information Processing Systems*, pages 3592–3602. 134

Pedreschi, D., Giannotti, F., Guidotti, R., Monreale, A., Ruggieri, S., and Turini, F. (2019). Meaningful explanations of black box ai decision systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9780–9784. 125

Pei, S., Hu, Q., and Chen, C. (2016). Multivariate decision trees with monotonicity constraints. *Know.-Based Syst.*, **112**(C), 14–25. 47, 53

Perron, L. and Furnon, V. (2019). Google: Or-tools. `https://developers.google.com/optimization/`. 114

Pessach, D. and Shmueli, E. (2022). A review on fairness in machine learning. *ACM Computing Surveys (CSUR)*, **55**(3), 1–44. 72, 73, 74

Piltaver, R., Luštrek, M., Gams, M., and Martinčić-Ipšić, S. (2016). What makes classification trees comprehensible? *Expert Syst. Appl.*, **62**(C), 333–346. 37, 42, 45, 46, 61, 90

Potharst, R. and Feelders, A. J. (2002). Classification trees for problems with monotonicity constraints. *SIGKDD Explor. Newsl.*, **4**(1), 1–10. 48, 56, 62

Qiu, C., Jiang, L., and Li, C. (2017). Randomly selected decision tree for test-cost sensitive learning. *Appl. Soft Comput.*, **53**(C), 27–33. 49

Quinlan, J. and Cameron-Jones, R. (1995). Oversearching and layered search in empirical learning. In *14th International Joint Conference on Artificial Intelligence 95*. 121, 144

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, **1**(1), 81–106. 13, 144

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 13, 90

Quinlan, J. R. and Rivest, R. L. (1989). Inferring decision trees using the minimum description lenght principle. *Information and computation*, **80**(3), 227–248. 42, 56, 58, 62

Raff, E., Sylvester, J., and Mills, S. (2018). Fair forests: Regularized tree induction to minimize model bias. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, pages 243–250, New York, NY, USA. ACM Press. 52, 53, 56, 76

Read, S. H., Rosella, L. C., Berger, H., Feig, D. S., Fleming, K., Kaul, P., Ray, J. G., Shah, B. R., and Lipscombe, L. L. (2021). Diabetes after pregnancy: a study protocol for the derivation and validation of a risk prediction model for 5-year risk of diabetes following pregnancy. *Diagnostic and Prognostic Research*, **5**(1), 1–8. 106

Ren, F., Ding, X., Zheng, M., Korzinkin, M., Cai, X., Zhu, W., Mantsyzov, A., Aliper, A., Aladinskiy, V., Cao, Z., *et al.* (2022). Alphafold accelerates artificial intelligence powered drug discovery: Efficient discovery of a novel cyclin-dependent kinase 20 (cdk20) small molecule inhibitor. *arXiv preprint arXiv:2201.09647*. 5

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Model-agnostic interpretability of machine learning. In *ICML Workshop on Human Interpretability in Machine Learning*, WHI '16, Stockholm, Sweden. 38, 89, 124

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*. 125, 134

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., *et al.* (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, **115**(3), 211–252. 73

Safavian, S. R. and Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, **21**(3), 660–674. 12, 40

Saravanakumar, K. K. (2020). The impossibility theorem of machine fairness–a causal perspective. *arXiv preprint arXiv:2007.06024*. 76

Sato, M. and Tsukimoto, H. (2001). Rule extraction from neural networks via decision tree induction. In *International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, volume 3 of *IJCNN'01*, pages 1870–1875. IEEE. 44

Saxena, N. A., Huang, K., DeFilippis, E., Radanovic, G., Parkes, D. C., and Liu, Y. (2019). How do fairness definitions fare? examining public attitudes towards algorithmic definitions of fairness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 99–106. 74

Schetinin, V., Fieldsend, J. E., Partridge, D., Coats, T. J., Krzanowski, W. J., Everson, R. M., Bailey, T. C., and Hernandez, A. (2007). Confident interpretation of bayesian decision tree ensembles for clinical applications. *Trans. Info. Tech. Biomed.*, **11**(3), 312–319. 57, 60

Sener, O. and Koltun, V. (2018). Multi-task learning as multi-objective optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 525–536. 132, 141

Sethi, I. K. and Sarvarayudu, G. P. R. (1982). Hierarchical classifier design using mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.*, **4**(4), 441–445. 54, 56

Shaharanee, I. N. M., Hadzic, F., and Dillon, T. S. (2011). Interestingness measures for association rules based on statistical validity. *Knowledge-Based Systems*, **24**(3), 386–392. 92

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press. 9

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. (2020). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, page 180–186, New York, NY, USA. 125

Smith, J. W., Everhart, J. E., Dickson, W., Knowler, W. C., and Johannes, R. S. (1988). Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *annual symposium on computer application in medical care*. American Medical Informatics Association. 106

Souris, A., Bhattacharya, A., and Pati, D. (2018). The soft multivariate truncated normal distribution. *arXiv preprint arXiv:1807.09155*. 127

Struyf, J. and Džeroski, S. (2006). Constraint based induction of multi-objective regression trees. In *Proceedings of the 4th International Conference on Knowledge Discovery in Inductive Databases*, KDID'05, pages 222–233, Berlin, Heidelberg. Springer-Verlag. 57, 68

Struyf, J. and Džeroski, S. (2007). Clustering trees with instance level constraints. In *Proceedings of the 18th European Conference on Machine Learning*, ECML '07, pages 359–370, Berlin, Heidelberg. Springer-Verlag. 41, 54, 56

Suresh, H. and Guttag, J. V. (2019). A framework for understanding unintended consequences of machine learning. *arXiv preprint arXiv:1901.10002*, **2**, 8. 73, 74

Sweeney, L. (2002). K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, **10**(5), 557–570. 50, 56

Tan, M. (1993). Cost-sensitive learning of classification knowledge and its applications in robotics. *Mach. Learn.*, **13**(1), 7–33. 49, 56

Teng, Z. and Du, W. (2007). A hybrid multi-group privacy-preserving approach for building decision trees. In *Proceedings of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD'07, pages 296–307, Berlin, Heidelberg. Springer-Verlag. 50, 56

Nanfack, G., Delchevalerie, V., and Frénay, B. (2021a). Boundary-based fairness constraints in decision trees and random forests. In *The 29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, ESANN'21. 40, 47, 56, 71, 84

Nanfack, G., Temple, P., and Frénay, B. (2021b). Global explanations with decision rules: a co-learning approach. In *The 37th Conference on Uncertainty in Artificial Intelligence*, UAI'21. 123, 130, 131, 133

Nanfack, G., Temple, P., and Frénay, B. (2022a). Constraint enforcement on decision trees: a survey. *ACM Computing Surveys (CSUR)*. 36, 38, 41, 42, 53, 56, 61, 62, 63, 146

Nanfack, G., Temple, P., and Frénay, B. (2022b). Learning customised decision trees under domain-knowledge constraints. *Under Review for Pattern Recognition*. 87, 109, 119, 120

Tina Binesh, M. and Sydney, M. (2018). *Toronto Notes for Medical Students: comprehensive medical reference and review for MCCQE and USMLE II*. Toronto Notes 2018, Toronto, Ontario, Canada, 34th ed edition. 105

Tjortjis, C. and Keane, J. (2002). T3: A classification algorithm for data mining. In *Proceedings of the Third International Conference on Intelligent Data Engineering and Automated Learning*, IDEAL '02, pages 50–55, Berlin, Heidelberg. Springer-Verlag. 44, 47, 57, 59

Torres, P., Riaño, D., and López-Vallverdú, J. A. (2011). Inducing decision trees from medical decision processes. In *Proceedings of the ECAI 2010 Conference on Knowledge Representation for Health-care*, KR4HC'10, pages 40–55, Berlin, Heidelberg. Springer-Verlag. 50

Turney, P. D. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *J. Artif. Int. Res.*, **2**(1), 369–409. 49, 57, 59

Tzirakis, P. and Tjortjis, C. (2017). T3c: Improving a decision tree classification algorithm's interval splits on continuous attributes. *Adv. Data Anal. Classif.*, **11**(2), 353–370. 44, 57, 59

Vaghashia, H. and Ganatra, A. (2015). A survey: privacy preservation techniques in data mining. *International Journal of Computer Applications*, **119**(4), 20–26. 50

Vaidya, J., Clifton, C., Kantarcioglu, M., and Patterson, A. S. (2008). Privacy-preserving decision trees over vertically partitioned data. *ACM Trans. Knowl. Discov. Data*, **2**(3), 14:1–14:27. 51, 56

Verbakel, J. Y., Lemiengre, M. B., De Burghgraeve, T., De Sutter, A., Aertgeerts, B., Bullens, D. M., Shinkins, B., Van den Bruel, A., and Buntinx, F. (2015). Validating a decision tree for serious infection: diagnostic accuracy in acutely ill children in ambulatory care. *BMJ open*, **5**(8). 92, 116, 117

Verhaeghe, H., Nijssen, S., Pesant, G., Quimper, C.-G., and Schaus, P. (2019). Learning optimal decision trees using constraint programming. In *The 25th International Conference on Principles and Practice of Constraint Programming (CP'19)*. 45, 57, 62, 63, 91, 92, 99, 113, 114

Verwer, S. and Zhang, Y. (2017). Learning decision trees with flexible constraints and objectives using integer optimization. In *14th International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, CPAIOR'17, pages 94–103, Cham. Springer-Verlag. 57, 59, 65

Verwer, S. and Zhang, Y. (2019). Learning optimal classification trees using a binary linear program formulation. In *33rd AAAI Conference on Artificial Intelligence*, AAAI '19. AAAI Press. 45, 57, 59, 62, 63, 64, 65, 89, 91, 92, 99, 112, 113, 114, 115, 118, 124, 143

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Walczak, M., Pfrommer, J., Pick, A., *et al.* (2021). Informed machine learning-a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge & Data Engineering*, pages 1–1. 41

Wagstaff, K., Cardie, C., Rogers, S., and Schroedl, S. (2001a). Constrained k-means clustering with background knowledge. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML'01, pages 577–584. Morgan Kaufmann. 54

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., *et al.* (2001b). Constrained k-means clustering with background knowledge. In *18th International Conference on Machine Learning*. 103

Wang, N., Li, J., Liu, Y., Zhu, J., Su, J., and Peng, C. (2018). Accurate decision tree with cost constraints. In *First International Conference on Advanced Hybrid Information Processing*, ADHIP '17, pages 154–165, Cham. Springer-Verlag. 49, 56

Wenger, N. K., Speroff, L., and Packard, B. (1993). Cardiovascular health and disease in women. *New England Journal of Medicine*, **329**(4), 247–256. 109

Wick, M., Panda, S., and Tristan, J.-B. (2019). Unlocking fairness: a trade-off revisited. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8783–8792. 84

Wolf, L., Galanti, T., and Hazan, T. (2019). A formal approach to explainability. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, page 255–261, New York, NY, USA. Association for Computing Machinery. 125

Wu, C.-C., Chen, Y.-L., Liu, Y.-H., and Yang, X.-Y. (2016). Decision tree induction with a constrained number of leaf nodes. *Applied Intelligence*, **45**(3), 673–685. 56, 58

Wu, M., Hughes, M. C., Parbhoo, S., Zazzi, M., Roth, V., and Doshi-Velez, F. (2018). Beyond sparsity: Tree regularization of deep models for interpretability. In *AAAI Conference on Artificial Intelligence*. 125

Wu, M., Parbhoo, S., Hughes, M. C., Kindle, R., Celi, L. A., Zazzi, M., Roth, V., and Doshi-Velez, F. (2020). Regional tree regularization for interpretability in deep neural networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 6413–6421. 125, 133

Wu, Y., Tjelmeland, H., and West, M. (2007). Bayesian cart: Prior specification and posterior simulation. *Journal of Computational and Graphical Statistics*, **16**(1), 44–66. 57

Xu, T., Chongxuan, L., Zhu, J., and Zhang, B. (2019). Multi-objects generation with amortized structural regularization. In *Advances in Neural Information Processing Systems*, pages 6619–6629. 130

Yang, C., Rangarajan, A., and Ranka, S. (2018). Global model interpretation via recursive partitioning. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1563–1570. IEEE. 43, 56, 66

Yang, H., Rudin, C., and Seltzer, M. (2017). Scalable bayesian rule lists. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 3921–3930. 124

Yang, W., Lorch, L., Graule, M. A., Lakkaraju, H., and Doshi-Velez, F. (2020). Incorporating interpretable output constraints in bayesian neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 12721–12731. 81, 105, 112, 148

You, S., Ding, D., Canini, K., Pfeifer, J., and Gupta, M. R. (2017). Deep lattice networks and partial monotonic functions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 2985–2993, USA. Curran Associates Inc. 52

Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. 73

Zafar, M. B., Valera, I., Rogriguez, M. G., and Gummadi, K. P. (2017). Fairness Constraints: Mechanisms for Fair Classification. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 962–970. PMLR. 52, 53, 75

Zafar, M. B., Valera, I., Gomez-Rodriguez, M., and Gummadi, K. P. (2019). Fairness constraints: A flexible approach for fair classification. *The Journal of Machine Learning Research*, **20**(1), 2737–2778. 75, 77, 79, 84, 142, 143

Zaman, A. N. K., Obimbo, C., and Dara, R. A. (2016). A novel differential privacy approach that enhances classification accuracy. In *Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering*, C3S2E '16, pages 79–84, New York, NY, USA. ACM. 51, 56

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*. 51, 74

Zhang, Q., Yang, Y., Ma, H., and Wu, Y. N. (2019). Interpreting cnns via decision trees. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6261–6270. 139

Zhang, W. and Ntoutsi, E. (2019). Faht: an adaptive fairness-aware decision tree classifier. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1480–1486. 76, 82, 142, 143

Zhang, Y. and Long, Q. (2021). Assessing fairness in the presence of missing data. *Advances in neural information processing systems*, **34**, 16007–16019. 52

Zhang, Y., Xiang, T., Hospedales, T. M., and Lu, H. (2018). Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4320–4328. 126

Zilke, J. R., Mencía, E. L., and Janssen, F. (2016). Deepred–rule extraction from deep neural networks. In *19th International Conference on Discovery Science*, DS '16, pages 457–473, Cham. Springer-Verlag. 44, 56, 58, 66