Journal of Machine Learning Research 1 (2023) 1-7 Submitted 4/00; Published 10/00

ODTlearn: A Package for Learning Optimal Decision Trees for Prediction and Prescription

Patrick Vossler* PVOSSLER@USC.EDU
Sina Aghaei SAGHAEI@USC.EDU
Nathan Justin NJUSTIN@USC.EDU
Nathanael Jo NATHANAEL.JO@GMAIL.COM
Andrés Gómez GOMEZAND@USC.EDU
Phebe Vayanos PHEBE.VAYANOS@USC.EDU
University of Southern California, Center for AI in Society, Los Angeles, CA 90089

Editor:

Abstract

`ODTLearn` is an open-source Python package that provides methods for learning optimal decision trees for high-stakes predictive and prescriptive tasks based on the mixed-integer optimization (MIO) framework proposed in (Aghaei et al., 2019) and several of its extensions. The current version of the package provides implementations for learning optimal classification trees, optimal fair classification trees, optimal classification trees robust to distribution shifts, and optimal prescriptive trees from observational data. We have designed the package to be easy to maintain and extend as new optimal decision tree problem classes, reformulation strategies, and solution algorithms are introduced. To this end, the package follows object-oriented design principles and supports both commercial (Gurobi) and open source (COIN-OR branch and cut) solvers. The package documentation and an extensive user guide can be found at `https://d3m-research-group.github.io/odtlearn/`. Additionally, users can view the package source code and submit feature requests and bug reports by visiting `https://github.com/D3M-Research-Group/odtlearn`.

Keywords: Mixed-integer optimization, prescriptive trees, classification trees, distribution shifts, fair classification trees, robust classification trees, open source software.

1. Introduction

Automated data-driven predictive and prescriptive methods are increasingly being used in high-stakes domains to inform and support decision-making. In such settings, these machine learning (ML) and artificial intelligence (AI) tools should be: (a) *accurate* (to minimize erroneous predictions/prescriptions that may negatively affect the populations on which they are deployed), (b) *interpretable* (so that predictions and decisions are transparent, accountable, and easy to audit), (c) *flexible* (i.e., possible to easily augment with domain specific constraints such as capacity and/or fairness constraints), and (d) *robust* (to ensure high-quality solutions even under adversarial shifts between training and deployment data).

Despite their popularity, decision trees (Breiman et al., 1984) are not necessarily well suited for data-driven decision-making in high-stakes domains. While the structure of de-

*. Corresponding Author

cision trees makes them easy to interpret, they are typically constructed using heuristics and may yield suboptimal solutions. Furthermore, with heuristic-based decision trees, it is not immediately apparent how to incorporate relevant side information into the tree construction process or how to make a tree robust to adversarial shifts between training and deployment data. Optimal decision trees retain the interpretability of heuristic decision trees while still being flexible enough to model the types of problems decision-makers face and provide optimal solutions that decision-makers can trust.

Our `ODTlearn` Python package provides methods for fitting provably optimal decision trees using mixed-integer optimization (MIO) for various problem types and settings commonly encountered by practitioners in high-stakes settings. The modeling methods provided in `ODTlearn` build upon the modeling and solution paradigm proposed by Aghaei et al. (2021), which is significantly faster and provides better out-of-sample performance than previous MIO-based algorithms (see Section 2 for additional discussion). This approach generalizes beyond standard classification problems to problems involving imbalanced datasets (e.g., by optimizing weighted accuracy or worst-case accuracy, by constraining recall or precision, or by balancing sensitivity and specificity). In addition to the core algorithm, we implement its generalization to learn optimal fair decision trees that optimize accuracy while satisfying arbitrary domain specific fairness constraints such as statistical parity, conditional statistical parity, or equalized odds as proposed in (Jo et al., 2022). These were show to outperform some of the most popular (heuristic-based) algorithms for learning fair trees. For users deploying decision trees in settings with potential distribution shifts between training and testing, we implement the method proposed in (Justin et al., 2021) for learning optimal robust classification trees. Finally, we implement the method in (Jo et al., 2022) for learning optimal prescriptive trees from observational data. This framework can be used to design treatment assignment policies in the form of decision trees, being highly interpretable while offering a tunable degree of personalization. Importantly, the learned trees can also be constrained to satisfy domain specific requirements such as budget constraints (e.g., limited amount of treatments) or fairness constraints (e.g., conditional statistical parity in allocation or in outcomes). Our `ODTlearn` Python package provides methods for fitting provably optimal decision trees using mixed-integer optimization (MIO) for various problem types and settings commonly encountered by practitioners in high-stakes settings. The modeling methods provided in `ODTlearn` build upon the modeling and solution paradigm proposed by Aghaei et al. (2021), which is significantly faster and provides better out-of-sample performance than previous MIO-based algorithms (see Section 2 for additional discussion). This approach generalizes beyond standard classification problems to problems involving imbalanced datasets (e.g., by optimizing weighted accuracy or worst-case accuracy, by constraining recall or precision, or by balancing sensitivity and specificity). In addition to the core algorithm, we implement its generalization to learn optimal fair decision trees that optimize accuracy while satisfying arbitrary domain specific fairness constraints such as statistical parity, conditional statistical parity, or equalized odds as proposed in (Jo et al., 2022). These were show to outperform some of the most popular (heuristic-based) algorithms for learning fair trees. For users deploying decision trees in settings with potential distribution shifts between training and testing, we implement the method proposed in (Justin et al., 2021) for learning optimal robust classification trees. Finally, we implement the method in (Jo et al., 2022) for learning optimal

prescriptive trees from observational data. This framework can be used to design treatment assignment policies in the form of decision trees, being highly interpretable while offering a tunable degree of personalization. Importantly, the learned trees can also be constrained to satisfy domain specific requirements such as budget constraints (e.g., limited amount of treatments) or fairness constraints (e.g., conditional statistical parity in allocation).

The remainder of this paper is organized as follows. Section 2 compares `ODTlearn` to related packages. Its architecture and a usage example are introduced in Section 3. Finally, the quality practices under which the package is developed are described in Section 4.

2. Comparison to Related Software

The most related software package is provided by Interpretable AI (Interpretable AI, 2022). This is a proprietary software package written in Julia (Bezanson et al., 2017) that provides methods for learning optimal classification trees and prescriptive trees using MIO based on papers by Bertsimas and Dunn (2017) and Amram et al. (2022), respectively. It features a Python wrapper that allows users to call these methods from Python. Unlike `ODTlearn`, the algorithms in the Interpretable AI package do not provide functionality for incorporating side constraints such as fairness constraints nor to handle settings involving a distribution shift in the data. Additionally, the MIO formulation in the `ODTlearn` package has been shown to be 50 times faster than the MIO formulation used in Interpretable AI (Aghaei et al., 2021). Furthermore, for classification problems and prescriptive problems, the methods in `ODTlearn` have been shown in experiments to be 13% and 13.6% more accurate on out-of-sample observations, respectively (Aghaei et al., 2021; Jo et al., 2021).

`PyDL8.5` (Aglin et al., 2021) is an open source Python package that implements the DL8.5 (Aglin et al., 2020) algorithm for learning optimal decision trees through the use of itemset mining techniques. The package provides methods for learning optimal classification trees, however their method is unable to incorporate side constraints.

Finally, while packages such as `scikit-learn` (Pedregosa et al., 2011) provide high-quality implementations of heuristic-based methods for learning decision trees, these methods lack the modeling flexibility and optimality guarantees of MIO-based methods.

3. Software Architecture and Usage Example

The software architecture of `ODTlearn` is motivated by the optimal decision tree literature in which researchers have recently proposed numerous new problem classes, reformulation strategies, and solution approaches. With this in mind, we have created a class structure for our package that follows the SOLID principles of object-oriented programming for developing software (Martin, 2003). These principles emphasize structuring classes that are easy to maintain and extend. Figure 1 shows the inheritance diagram in `ODTlearn`. The classes in `ODTlearn` are derived from our abstract base class `OptimalDecisionTree`. This class provides a standardized interface for the two types of trees currently supported (classification and prescription) while keeping their implementation details separate from one another. `OptimalClassificationTree` and `OptimalPrescriptiveTree` extend `OptimalDecisionTree` by specifying problem-specific methods for traversing and visualizing the decision tree. The structure prevents unnecessary code duplication within each child class. Next, the chil-

dren of `OptimalClassificationTree` and `OptimalPrescriptiveTree` implement methods for creating the decision variables, the constraints, and the objective function necessary for constructing the optimization problem of interest. The separate classes for each of the variations of the MIO formulations ensure that any changes to the structure of one problem formulation do not affect any of the other problem formulations. Finally, the classes in the third and fourth levels of Figure 1 implement user-facing methods such as `fit` and `predict`. Thus, our adherence to the SOLID principles ensures that researchers and practitioners building upon `ODTlearn` can easily augment it with more features (e.g., different objectives or additional constraints) or even build new types of trees altogether.

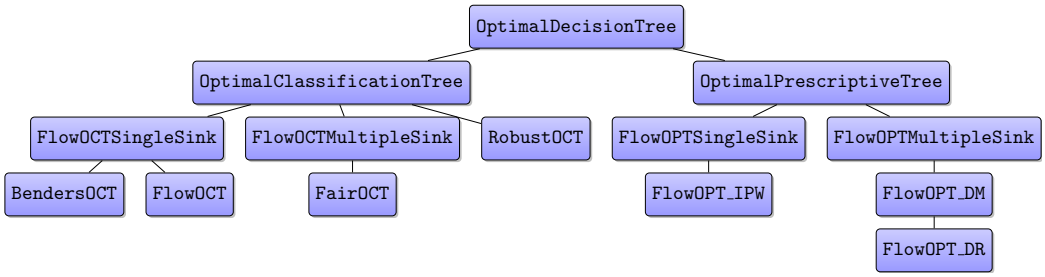

Figure 1: Inheritance Diagram for `ODTlearn`.

Figure 2 provides a code snippet demonstrating how to use the `ODTlearn` API to fit a fair optimal classification tree. This is achieved through the API using the familiar fit-predict structure. Once an optimal decision tree has been learned, users can employ the build in `plot_tree` function to visualize the tree. Finally, we note that while the fair optimal classification tree in the example uses the Gurobi solver (Gurobi Optimization, LLC, 2023), users can also use the open source COIN-OR branch and cut solver (Forrest et al., 2023) because `ODTlearn` builds upon the `python-mip` (Santos and Toffolo, 2020) solver interface.

```
from odtlearn.datasets import fairness_example
from odtlearn.fair_oct import FairOCT
import matplotlib.pyplot as plt

X, y, protect_feat, legit_factor =
    fairness_example()
fcl = FairOCT(solver="gurobi",
    positive_class=1, depth=2, _lambda=0.01,
fairness_type="SP", fairness_bound=1)
fcl.fit(X, y, protect_feat, legit_factor)
fcl_predictions = fcl.predict(X)

fcl.plot_tree()
plt.show()
```

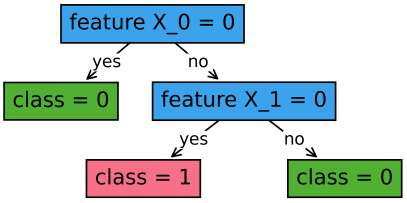

Figure 2: Code for fitting an optimal fair classification tree on a toy data set and a visualization of the learned classification tree.

4. Development

Releases of the ODTlearn package are available via PyPI at `https://pypi.org/project/odtlearn/1.0.0/`. The package source code and documentation are hosted on GitHub (`https://github.com/D3M-Research-Group/odtlearn`). Collaboration in the form of discussions, feature requests, or bug reports is made possible through the GitHub issue and pull request workflow. We have implemented continuous integration through GitHub Actions to ensure backward compatibility and quickly identify any code regressions. Our documentation includes installation instructions, a user guide, an API reference, and downloadable example notebooks demonstrating each of the classification methods implemented in the package. The documentation is hosted via GitHub pages at `https://d3m-research-group.github.io/odtlearn`. The package is distributed under the GPL-3.0 license and makes use of several core libraries within Python's scientific computing ecosystem: `scikit-learn` (Pedregosa et al., 2011), `numpy` (Harris et al., 2020), and `pandas` (McKinney et al., 2010).

Acknowledgments

P. Vayanos and N. Justin are funded in part by the National Science Foundation under CAREER award number 2046230. She is grateful for this support. P. Vayanos and P. Vossler acknowledge support from the USC Zumberge Special Solicitation – Epidemic & Virus Related Research and Development award. N. Jo acknowledges support from the Epstein Institute at the University of Southern California. A. Gómez is funded in part by the National Science Foundation under grant 2152777. N. Justin is funded in part by the National Science Foundation under the Graduate Research Fellowship Program.

References

Sina Aghaei, Mohammad Javad Azizi, and Phebe Vayanos. 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, 2019.

Sina Aghaei, Andrés Gómez, and Phebe Vayanos. Strong optimal classification trees. *arXiv preprint arXiv:2103.15965*, 2021.

Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3146–3153, 2020.

Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. Pydl8. 5: a library for learning optimal decision trees. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 5222–5224, 2021.

Maxime Amram, Jack Dunn, and Ying Daisy Zhuo. Optimal policy trees. *Machine Learning*, 111(7):2741–2768, 2022.

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

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. doi: 10.1137/141000671. URL `https://epubs.siam.org/doi/10.1137/141000671`.

Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 1984.

John Forrest, Ted Ralphs, Haroldo Gambini Santos, Stefan Vigerske, John Forrest, Lou Hafer, Bjarni Kristjansson, jpfasano, EdwinStraver, Miles Lubin, Jan-Willem, rlougee, jpgoncal1, Samuel Brito, h-i Gassmann, Cristina, Matthew Saltzman, tosttost, Bruno Pitrus, Fumiaki MATSUSHIMA, and to st. coin-or/cbc: Release releases/2.10.10, April 2023. URL `https://doi.org/10.5281/zenodo.7843975`.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL `https://www.gurobi.com`.

Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with Numpy. *Nature*, 585(7825):357–362, 2020.

LLC Interpretable AI. Interpretable AI documentation, 2022. URL `https://www.interpretable.ai`.

Nathanael Jo, Sina Aghaei, Andrés Gómez, and Phebe Vayanos. Learning optimal prescriptive trees from observational data. *arXiv preprint arXiv:2108.13628*, 2021.

Nathanael Jo, Sina Aghaei, Jack Benson, Andrés Gómez, and Phebe Vayanos. Learning optimal fair classification trees. *arXiv preprint arXiv:2201.09932*, 2022.

Nathan Justin, Sina Aghaei, Andres Gomez, and Phebe Vayanos. Optimal robust classification trees. In *The AAAI-22 Workshop on Adversarial Machine Learning and Beyond*, 2021.

Robert Cecil Martin. *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.

Wes McKinney et al. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Haroldo G Santos and T Toffolo. Mixed integer linear programming with Python. *Accessed: Apr*, 2020.