Learning Accurate and Interpretable Decision Trees (Extended Abstract)*

Maria-Florina Balcan[1] , **Dravyansh Sharma**[2]
[1]Carnegie Mellon University
[2]Toyota Technological Institute at Chicago
ninamf@cs.cmu.edu, dravy@ttic.edu

Abstract

Decision trees are a popular tool in machine learning and yield easy-to-understand models. Several techniques have been proposed in the literature for learning a decision tree classifier, with different techniques working well for data from different domains. In this work, we develop a data-driven approach to design decision tree learning algorithms given repeated access to data from the same domain. We study multiple formulations covering different aspects and popular techniques for learning decision trees. We propose novel parameterized classes of node splitting criteria in top-down algorithms, which interpolate between popularly used entropy and Gini impurity based criteria, and provide theoretical bounds on the number of samples needed to learn the splitting function appropriate for the data at hand. We also study the sample complexity of tuning prior parameters in Bayesian decision tree learning, and extend our results to decision tree regression. We further consider the problem of tuning hyperparameters in pruning the decision tree for classical pruning algorithms including min-cost complexity pruning. We also study the interpretability of the learned decision trees and introduce a data-driven approach for optimizing the explainability versus accuracy trade-off using decision trees. Finally, we demonstrate the significance of our approach on real world datasets by learning data-specific decision trees which are simultaneously more accurate and interpretable.

1 Introduction

Decision trees are ubiquitous, with applications in operations research, management science, data mining, and machine learning. They are easy to use and understand models that explicitly include the decision rules used in making predictions. Each decision rule is a simple comparsion of a real-valued attribute to a threshold or a categorical attribute against a candidate set of values. Given their remarkable simplicity, decision trees are widely preferred in applications where it is important to justify algorithmic decisions with intuitive explanations [Rudin, 2018]. However, decades of research on decision trees has resulted in a large suite of candidate approaches for building decision trees [Breiman *et al.*, 1984; Mingers, 1987; Quinlan, 1993; Quinlan, 1996; Kearns and Mansour, 1996; Mansour, 1997; Maimon and Rokach, 2014]. This raises an important question: how should one select the best approach to build a decision tree for the relevant problem domain?

Several empirical studies have been performed comparing various ways to build decision trees [Mingers, 1989b; Esposito *et al.*, 1997]. Current wisdom from the literature dictates that for any problem at hand, one needs a domain expert to try out, compare and tune various methods to build the best decision trees for any given problem domain. For instance, the popular Python library Scikit-learn [Pedregosa *et al.*, 2011] implements both Gini impurity and entropy as candidate 'splitting criteria' (a crucial component in building the decision trees top-down by deciding which node to split into child nodes), and yet theory suggests another promising candidate [Kearns and Mansour, 1996] that achieves smaller error bounds under the Weak Hypothesis Assumption[1]. It is therefore desirable to determine which approach works better for the data coming from a given domain. With sufficient data, can we automate this tedious manual process?

In this work we approach this crucial question, and propose ways to build more effective decision trees automatically. Our results show provable learning theoretic guarantees and select methods over larger search spaces than what human experts would typically explore. For example, instead of comparing a small finite number of splitting criteria, we examine learnability over continuously infinite parameterized families that yield more effective decision tree learning algorithms.

We consider the problem where the learner has access to multiple related datasets $D_1, \ldots, D_N$ coming from the same problem domain (given by a fixed but unknown distribution $\mathcal{D}$), and the goal is to design a decision tree learning algorithm that works well over the distribution $\mathcal{D}$ using as few datasets (N, the sample complexity) as possible. This algorithm design problem is typically formulated as the selection

*Full version appears in the proceedings of UAI 2024 [Balcan and Sharma, 2024], winner of Outstanding Student Paper Award.

[1]an *a priori* assumption on the target function. Roughly speaking, it means that the decision tree node functions are already slightly correlated with the target function.

of a hyperparameter from an infinite family. Typically finding the best hyperparameters even on a single problem sample is tedious and computationally intensive, so we would like to bound the number of samples over which we should optimize them, while learning parameters that generalize well over the distribution generating the problem samples. We take steps towards systematically unifying, automating and formalizing the process of designing decision tree learning algorithms, in a way that is adaptive to the data domain.

1.1 Our Contributions

We formulate the problem of designing a decision tree learning algorithm as a hyperparameter selection problem over multiple problem instances coming from the same domain. Under this formulation, we study the sample complexity, i.e. the number of problem instances needed to learn a provably good algorithm (hyperparameter) under the statistical learning setting (meaning problem instances are drawn from a fixed but unknown distribution) from several different design perspectives important in the construction of decision trees. A key technical challenge is the non-linearity of boundaries of the piecewise structured dual loss function.

- We introduce a novel family of node splitting criterion called (α, β)-Tsallis entropy criterion, which contains two tunable parameters, and includes several popular node splitting criteria from the literature including the entropy-based ID3/C4.5 [Quinlan, 1986; Quinlan, 1993] and Gini impurity based CART [Breiman *et al.*, 1984]. We bound the sample complexity of provably tuning these hyperparameters in top-down learning algorithms. We perform experiments to show the practical significance and effectiveness of tuning these hyperparameters on real world datasets.
- We further study tuning of parameters in Bayesian decision tree learning algorithms used in generating the prior distribution. We also study a parameterized family for node splitting for regression trees and bound the sample complexity of tuning the parameter.
- We next consider the problem of learning the pruning algorithm used in constructing the decision tree. We show how to tune parameters in popular algorithms including the complexity parameter $\tilde{\alpha}$ in the Minimal Cost-Complexity Pruning algorithm, and again obtain sample complexity bounds. We also study the sample complexity of tuning pessimistic error pruning methods, which are computationally faster.
- We consider the problem of optimizing the explainability-accuracy trade-off in the design of decision tree learning algorithms. Here we consider tuning splitting and pruning parameters simultaneously when growing a decision tree to size t and pruning it down to size $t' \leq t$, while minimizing an objective that incorporates explainability as well as accuracy. Our work is the first to study explainability from a data-driven design perspective.

We highlight the first couple of contributions above in this extended abstract. For further details on the other contributions, see the full version [Balcan and Sharma, 2024].

1.2 Related Work

Building and pruning decision trees. Typically, decision trees are built in two stages. First the tree is grown in a top-down fashion by successively 'splitting' existing nodes according to some *splitting criterion*. Numerous different methods to select which node to split and how to split have been proposed in the literature [Breiman *et al.*, 1984; Quinlan, 1993; Kearns and Mansour, 1996]. The second stage involves pruning the tree to avoid overfitting the training set, and again a variety of approaches are known [Breiman *et al.*, 1984; Mingers, 1987; Quinlan, 1987; Mansour, 1997]. Furthermore, empirical works suggest that the appropriate method to use depends on the data domain at hand [Mingers, 1989a; Mingers, 1989b]. The task of selecting the best method or tuning the hyperparameters for a method is left to domain expert. Recent work has developed techniques for computing the optimal decision trees by using branch-and-bound and dynamic programming based techniques [Hu *et al.*, 2019; Lin *et al.*, 2020; Demirović *et al.*, 2022]. The key idea is to reduce the search space by tracking bounds on the objective value. However, these approaches are computationally more expensive than the classical greedy methods.

*Data-driven algorithm design*is a recently introduced framework [Gupta and Roughgarden, 2016] for designing algorithms using machine learning in order to optimize performance over problems coming from a common problem domain (see [Balcan, 2020] for a survey). The basic premise is to design algorithms for typical inputs instead of worst-case inputs by examining repeated problem instances. In machine learning, this can be used to provably tune hyperparameters (Balcan et al. [2018a; 2018b; 2023b; 2023a; 2024b], Blum et al. [2021], Bartlett et al. [2022], Sharma et al. [2023; 2024; 2025]) as opposed to employing heuristics like grid search or random search [Bergstra and Bengio, 2012] for which formal global-optimality guarantees are typically not known. General techniques have been developed in previous works [Balcan *et al.*, 2024a] for providing the sample complexity of tuning a linear combination of variable selection policies in branch-and-bound, and special cases of "path-wise" node selection policies have been studied [Balcan *et al.*, 2021]. In contrast, our work provides new technical insights for node selection policies relevant for decision tree learning which do not satisfy the previously studied path-wise properties and involve a more challenging non-linear interpolation. Prior work obtains a general result for tree search without any path-wise assumptions, but still require a linear interpolation of selection policies.

2 Data-driven Top-down Learning

Let $[k]$ denote the set of integers $\{1, 2, \ldots, k\}$. A (supervised) classification problem is given by a labeled dataset $D = (X, y)$ over some input domain $X \in \mathcal{X}^n$ and $y \in \mathcal{Y}^n = [c]^n$ where c denotes the number of distinct classes or categories. Let $\mathcal{D}$ be a distribution over classification problems of size n. We will consider parameterized families of decision tree learning algorithms, parameterized by some parameter $\rho \in \mathcal{P} \subseteq \mathbb{R}^d$ and access to datasets $D_1, \ldots, D_N \sim \mathcal{D}^N$. We do not assume that individual data points (X_i, y_i) are i.i.d. in

Algorithm 1 Top-down decision tree learner ($\mathcal{F}$, g_ρ, t)

Input: Dataset $D = (X, y)$
Parameters: Node function class $\mathcal{F}$, splitting criterion $g_\rho \in \mathcal{G}_\mathcal{P}$, tree size t
Output: Decision tree T

Initialize T to a leaf node labeled by most frequent label y in D.
while T has at most t internal nodes **do**
  $l^*, f^* \leftarrow \text{argmin}_{l \in \text{leaves}(T), f \in \mathcal{F}} G_\rho(T_{l \to f})$
  $T \leftarrow T_{l^* \to f^*}$
return T

any dataset D_j.

We consider a finite *node function class* $\mathcal{F}$ consisting of boolean functions $\mathcal{X} \to \{0, 1\}$ which are used to label internal nodes in the decision tree, i.e. govern given any data point $x \in \mathcal{X}$ whether the left or right branch should be taken when classifying x using the decision tree. Any given data point $x \in \mathcal{X}$ corresponds to a unique leaf node determined by the node function evaluations at x along some unique root-to-leaf path. Each leaf node of the decision tree is labeled by a class in $[c]$. Given a dataset (X, y) this leaf label is typically set as the most common label for data points $x \in X$ which are mapped to the leaf node.

We denote by $T_{l \to f}$ the tree obtained by *splitting* the leaf node l, which corresponds to replacing it by an internal node labeled by f and creating two child leaf nodes. We consider a parameterized class of splitting criterion $\mathcal{G}_\mathcal{P}$ over some parameter space $\mathcal{P}$ consisting of functions $g_\rho : [0, 1]^c \to \mathbb{R}_{\geq 0}$ for $\rho \in \mathcal{P}$. The splitting criterion governs which leaf to be split next and which node function $f \in \mathcal{F}$ to be used when building the decision tree using a top-down learning algorithm which builds a decision tree by successively splitting nodes using g_ρ until the size equals input tree size t. More precisely, suppose $w(l)$ (the *weight* of leaf l) denotes the number of data points in X that map to leaf l, and suppose $p_i(l)$ denotes the fraction of data points labeled by $y = i \in [c]$ among those points that map to leaf l. The splitting function over tree T is given by

$$G_\rho(T) = \sum_{l \in \text{leaves}(T)} w(l) g_\rho \left(\{p_i(l)\}_{i=1}^c \right),$$

and we build the decision tree by successively splitting the leaf nodes using node function f which cause the maximum decrease in the splitting function. For example, the information gain criterion may be expressed using $g_\rho(\{p_i(l)\}_{i=1}^c) = -\sum_{i=1}^c p_i \log p_i$.

Algorithm 1 summarizes this well-known general paradigm. We denote the tree obtained by the top-down decision tree learner on dataset D as $T_{\mathcal{F},\rho,t}(D)$. We study the 0-1 loss of the resulting decision tree classifier. If $T(x) \in [c]$ denotes the prediction of tree T on $x \in \mathcal{X}$, we define the loss on dataset $D(X, y)$ as $L(T, D) := \frac{1}{n} \sum_{i=1}^n \mathbf{I}[T(X_i) \neq y_i]$, where $\mathbf{I}[\cdot]$ denotes the 0-1 valued indicator function.

2.1 Learning to Split Nodes

In this section, we study the sample complexity of learning the splitting criteria. Given a discrete probability distribution $P = \{p_i\}$ with $\sum_{i=1}^c p_i = 1$, we define (α, β)-Tsallis entropy as

$$g_{\alpha,\beta}^{\text{TSALLIS}}(P) := \frac{C}{\alpha - 1} \left(1 - \left(\sum_{i=1}^c p_i^\alpha \right)^\beta \right),$$

where C is a normalizing constant (does not affect Algorithm 1), $\alpha \in \mathbb{R}^+, \beta \in \mathbb{Z}^+$. $\beta = 1$ corresponds to standard Tsallis entropy [Tsallis, 1988]. For example, $\alpha = 2, \beta = 1$ corresponds to Gini impurity, $\alpha = \frac{1}{2}, \beta = 2$ corresponds to the Kearns and Mansour [1996] criterion (using which error ϵ can be achieved with trees of size $\text{poly}(1/\epsilon)$) and $\lim_{\alpha \to 1} g_{\alpha,1}^{\text{TSALLIS}}(P)$ yields the (Shannon) entropy criterion.

We consider $\alpha \in \mathbb{R}^+$ and $\beta \in [B]$ for some positive integer B, and observe that several previously studied splitting criteria can be readily obtained by setting appropriate values of parameters α, β. We consider the problem of tuning the parameters α, β simultaneously, given access to multiple problem instances (datasets) drawn from some distribution $\mathcal{D}$. The goal is to find parameters $\hat{\alpha}, \hat{\beta}$ based on the training samples, so that on a random $D \sim \mathcal{D}$, the expected loss $\mathbb{E}_{D \sim \mathcal{D}} L(T_{\mathcal{F},(\hat{\alpha},\hat{\beta}),t}, D)$ is minimized. We will bound the sample complexity of the ERM Empirical Risk Minimization (ERM) principle, which given N problem samples $D_1, \ldots, D_N$ computes parameters $\hat{\alpha}, \hat{\beta}$ such that

$$\hat{\alpha}, \hat{\beta} = \text{argmin}_{\alpha > 0, \beta \in [B]} \sum_{i=1}^N L(T_{\mathcal{F},(\alpha,\beta),t}, D_i).$$

We obtain the following guarantee on the sample complexity of learning a near-optimal splitting criterion. The overall argument involves an induction on the size t of the tree (which has appeared in several prior works including Megiddo [1978] and Balcan et el. [2018a; 2021]), coupled with a counting argument for upper bounding the number of parameter sub-intervals corresponding to different behaviors of Algorithm 1 given a parameter interval corresponding to a fixed partial tree corresponding to an intermediate stage of the algorithm.

Theorem 1. *Suppose $\alpha > 0$ and $\beta \in [B]$. For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over problem instances with n examples, $O(\frac{1}{\epsilon^2}(t(\log |\mathcal{F}| + \log t + c \log(B + c)) + \log \frac{1}{\delta}))$ samples drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1 - \delta$ over the draw of the samples, the parameters $\hat{\alpha}, \hat{\beta}$ learned by ERM over the sample have expected loss that is at most ϵ larger than the expected loss of the best parameters $\alpha^*, \beta^* = \text{argmin}_{\alpha > 0, \beta \geq 1} \mathbb{E}_{D \sim \mathcal{D}} L(T_{\mathcal{F},(\hat{\alpha},\hat{\beta}),t}, D)$ over $\mathcal{D}$. Here t is the size of the decision tree, $\mathcal{F}$ is the node function class used to label the nodes of the decision tree and c is the number of label classes.*

2.2 Experiments

We examine the significance of the novel splitting techniques and the importance of designing data-driven decision tree learning algorithms via hyperparameter tuning for various benchmark datasets from the UCI repository.

We first study the effect of choice of (α, β) parameters in the Tsallis entropy based splitting criterion. For each dataset,

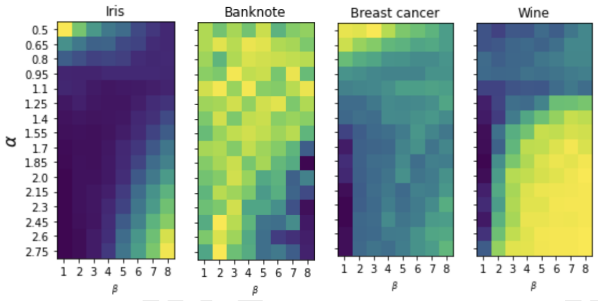

Figure 1: Average test accuracy (proportional to brightness, yellow is highest) of (α, β)-Tsallis entropy based splitting criterion as the parameters are varied, across datasets. We observe that different parameter settings work best for each dataset, highlighting the need to learn data-specific values.

we perform 5-fold cross validation for a large grid of parameters depicted in Figure 1 and measure the accuracy on held out test set consisting of 20% of the datapoints (i.e. training datasets are just random subsets of the 80% of the dataset used for learning the parameters). We implement a slight variant of Algorithm 1 which grows the tree to maximum depth of 5 (as opposed to a fixed size t). We do not use any pruning here. There is a remarkable difference in the optimal parameter settings for different datasets. Moreover, carefully chosen values of (α, β) significantly outperform standard heuristics like Gini impurity or entropy based splitting, or even specialized heuristics like [Kearns and Mansour, 1996] for which worst-case error guarantees (assuming weak learning) are known. This further underlines the significance of data-driven algorithm design for decision tree learning.

2.3 Bayesian Decision Tree Models

Several Bayesian approaches for building a decision tree have been proposed in the literature [Chipman *et al.*, 1998; Chipman *et al.*, 2002; Wu *et al.*, 2007]. The key idea is to specify a prior which induces a posterior distribution and a stochastic search is performed using Metropolis-Hastings algorithms to explore the posterior and find an effective tree. We will summarize the overall approach below and consider the problem of tuning parameters in the prior, which control the accuracy and size of the tree. Unlike most of prior research on data-driven algorithm design which study deterministic algorithms, we will analyze the learnability of parameters in a randomized algorithm. One notable exception is the study of random initialization of centers in k-center clustering via parameterized Llyod's families [Balcan *et al.*, 2018b].

σ, ϕ-Bayesian algorithm family. Let $F = (f_1, \ldots, f_t)$ denote the node functions at the nodes of the decision tree T. The prior $p(F, T)$ is specified using the relationship

$$p(F, T) = p(F|T)p(T).$$

We start with a tree T consisting of a single root node. For any node τ in T, it is split with probability $p_{\text{SPLIT}}(\tau) = \sigma(1 + d_\tau)^{-\phi}$, and if split, the process is repeated for the left and right children. Here d_τ denotes the depth of node τ, and σ, ϕ are hyperparameters. The size of generated tree is capped to some upper bound t. Intuitively, σ controls the size of the tree and ϕ controls its depth. At each node, the node function is selected uniformly at random from $\mathcal{F}$. This specifies the prior $p(T)$. The conjugate prior for the node functions $F = (f_1, \ldots, f_t)$ is given by the standard Dirichlet distribution of dimension $c - 1$ (recall c is the number of label classes) with parameter $a = (a_1, \ldots, a_c), a_i > 0$. Under this prior, the label predictions are given by

$$p(y \mid X, T) = \left(\frac{\Gamma(\sum_i a_i)}{\Pi_i \Gamma(a_i)}\right)^t \prod_{j=1}^{t} \frac{\Pi_i \Gamma(n_{ji} + a_i)}{\Gamma(n_j + \sum_i a_i)},$$

where $n_{ji} = \sum_k \mathbf{I}(y_{jk} = i)$ counts the number of datapoints with label i at node j, $n_j = \sum_i n_{ji}$ and $i = 1, \ldots, c$. a is usually set as the vector $(1, \ldots, 1)$ which corresponds to the uniform Dirichlet prior. Finally the stochastic search of the induced posterior is done using the Metropolis-Hastings (MH) algorithm for simulating a Markov chain [Chipman *et al.*, 1998]. Starting from a single root node, the initial tree T^0 is grown according to the prior $p(T)$. Then to construct T^{i+1} from T^i, a new tree T^* is constructed by splitting a random node using a random node function, pruning a random node, reassigning a node function or swapping the node functions of a parent and a child node. Then we set $T^{i+1} = T^*$ with probability $q(T^i, T^*)$ according to the posterior $p(y \mid X, T)$, or keep $T^{i+1} = T^i$ otherwise. The algorithm outputs the tree T^ω where ω is typically a fixed large number of iterations (say 10000) to ensure that the search space is explored sufficiently well.

Hyperparameter tuning. We consider the problem of tuning of prior hyperparameters σ, ϕ, to obtain the best expected performance of the algorithm. To this end, we define $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_{t-1}) \in [0, 1]^{t-1}$ as the randomness used in generating the tree T according to $p(T)$. Let $T_{\mathbf{z},\sigma,\phi}$ denote the resulting initial tree. Let $\mathbf{z}'$ denote the remaining randomness used in the selecting the random node function and the stochastic search, resulting in the final tree $T(T_{\mathbf{z},\sigma,\phi}, \mathbf{z}', \omega)$. Our goal is to learn the hyperparameters σ, ϕ which minimize the expected loss

$$\mathbb{E}_{\mathbf{z},\mathbf{z}',\mathcal{D}} L(T(T_{\mathbf{z},\sigma,\phi}, \mathbf{z}', \omega), D),$$

where $\mathcal{D}$ denotes the distribution according to which the data D is sampled, and L denotes the expected fraction of incorrect predictions by the learned Bayesian decision tree. ERM over a sample $D_1, \ldots, D_n \sim \mathcal{D}^n$ finds the parameters $\hat{\sigma}, \hat{\phi}$ which minimize the expected average loss $\frac{1}{n}\sum_{i=1}^n \mathbb{E}_{\mathbf{z},\mathbf{z}'} L(T(T_{\mathbf{z},\sigma,\phi}, \mathbf{z}', \omega), D_i)$ over the problem instances in the sample. It is not clear how to efficiently implement this procedure. However, we can bound its sample complexity and prove the following guarantee for learning a near-optimal prior for the Bayesian decision tree.

Theorem 2 (informal)**.** *Suppose $\sigma, \phi > 0$. The sample complexity of tuning the parameters (with at most ϵ error and high probability at least $1 - \delta$) in the σ, ϕ-Bayesian algorithm family is $O(\frac{1}{\epsilon^2}(\log t + \log \frac{1}{\delta}))$ given samples from an arbitrary distibution $\mathcal{D}$ (in the sense of Theorem 1).*

Acknowledgments

We thank Avrim Blum, Misha Khodak, Hedyeh Beyhaghi, Siddharth Prasad and Keegan Harris for helpful comments. This material is based on work supported by the National Science Foundation under grants CCF1910321, IIS 1901403, SES 1919453, ECCS-2216899, and ECCS-2216970; and the Defense Advanced Research Projects Agency under cooperative agreement HR00112020003.

References

[Balcan and Sharma, 2024] Maria-Florina Balcan and Dravyansh Sharma. Learning accurate and interpretable decision trees. In *Uncertainty in Artificial Intelligence (UAI)*, pages 288–307. PMLR, 2024.

[Balcan *et al.*, 2018a] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International Conference on Machine Learning (ICML)*, pages 344–353. PMLR, 2018.

[Balcan *et al.*, 2018b] Maria-Florina Balcan, Travis Dick, and Colin White. Data-driven clustering via parameterized Lloyd's families. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[Balcan *et al.*, 2021] Maria-Florina Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik. Sample complexity of tree search configuration: Cutting planes and beyond. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:4015–4027, 2021.

[Balcan *et al.*, 2023a] Maria-Florina Balcan, Avrim Blum, Dravyansh Sharma, and Hongyang Zhang. An analysis of robustness of non-Lipschitz networks. *Journal of Machine Learning Research (JMLR)*, 24(98):1–43, 2023.

[Balcan *et al.*, 2023b] Maria-Florina Balcan, Anh Nguyen, and Dravyansh Sharma. New bounds for hyperparameter tuning of regression problems across instances. *Advances in Neural Information Processing Systems*, 36, 2023.

[Balcan *et al.*, 2024a] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch: Generalization guarantees and limits of data-independent discretization. *Journal of the ACM*, 2024.

[Balcan *et al.*, 2024b] Maria-Florina Balcan, Christopher Seiler, and Dravyansh Sharma. Accelerating ERM for data-driven algorithm design using output-sensitive techniques. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:72648–72687, 2024.

[Balcan, 2020] Maria-Florina Balcan. Data-Driven Algorithm Design (book chapter). In *Beyond Worst Case Analysis of Algorithms, Tim Roughgarden (Ed)*. Cambridge University Press, 2020.

[Bartlett *et al.*, 2022] Peter Bartlett, Piotr Indyk, and Tal Wagner. Generalization bounds for data-driven numerical linear algebra. In *Conference on Learning Theory (COLT)*, pages 2013–2040. PMLR, 2022.

[Bergstra and Bengio, 2012] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research (JMLR)*, 13(2), 2012.

[Blum *et al.*, 2021] Avrim Blum, Chen Dan, and Saeed Seddighin. Learning complexity of simulated annealing. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1540–1548. PMLR, 2021.

[Breiman *et al.*, 1984] Leo Breiman, JH Friedman, RA Olshen, and CJ Stone. *Classification and regression trees. Statistics/probability series*. Wadsworth Publishing Company, 1984.

[Chipman *et al.*, 1998] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian CART model search. *Journal of the American Statistical Association (JASA)*, 93(443):935–948, 1998.

[Chipman *et al.*, 2002] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian treed models. *Machine Learning*, 48:299–320, 2002.

[Demirović *et al.*, 2022] Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J Stuckey. Murtree: Optimal decision trees via dynamic programming and search. *Journal of Machine Learning Research (JMLR)*, 23(1):1169–1215, 2022.

[Esposito *et al.*, 1997] Floriana Esposito, Donato Malerba, Giovanni Semeraro, and J Kay. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 19(5):476–491, 1997.

[Gupta and Roughgarden, 2016] Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. In *Innovations in Theoretical Computer Science (ITCS)*, pages 123–134, 2016.

[Hu *et al.*, 2019] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[Kearns and Mansour, 1996] Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Symposium on Theory of Computing (STOC)*, pages 459–468, 1996.

[Lin *et al.*, 2020] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning (ICML)*. PMLR, 2020.

[Maimon and Rokach, 2014] Oded Z Maimon and Lior Rokach. *Data mining with decision trees: theory and applications*, volume 81. World scientific, 2014.

[Mansour, 1997] Yishay Mansour. Pessimistic decision tree pruning based on tree size. In *International Conference on Machine Learning (ICML)*, pages 195–201, 1997.

[Megiddo, 1978] Nimrod Megiddo. Combinatorial optimization with rational objective functions. In *Symposium on Theory of Computing (STOC)*, pages 1–12, 1978.

[Mingers, 1987] John Mingers. Expert systems—rule induction with statistical data. *Journal of the Operational Research Society (JORS)*, 38:39–47, 1987.

[Mingers, 1989a] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 1989.

[Mingers, 1989b] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine learning*, 3:319–342, 1989.

[Pedregosa *et al.*, 2011] 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 (JMLR)*, 12:2825–2830, 2011.

[Quinlan, 1986] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Quinlan, 1987] J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies (IJMMS)*, 27(3):221–234, 1987.

[Quinlan, 1993] J Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[Quinlan, 1996] J. Ross Quinlan. Learning decision tree classifiers. *ACM Computing Surveys (CSUR)*, 28(1):71–72, 1996.

[Rudin, 2018] Cynthia Rudin. Please stop explaining black box models for high stakes decisions. *Stat*, 1050:26, 2018.

[Sharma and Jones, 2023] Dravyansh Sharma and Maxwell Jones. Efficiently learning the graph for semi-supervised learning. In *Uncertainty in Artificial Intelligence*, pages 1900–1910. PMLR, 2023.

[Sharma and Suggala, 2025] Dravyansh Sharma and Arun Suggala. Offline-to-online hyperparameter transfer for stochastic bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 2025.

[Sharma, 2024] Dravyansh Sharma. No internal regret with non-convex loss functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14919–14927, 2024.

[Tsallis, 1988] Constantino Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics*, 52:479–487, 1988.

[Wu *et al.*, 2007] Yuhong Wu, Håkon Tjelmeland, and Mike West. Bayesian CART: Prior specification and posterior simulation. *Journal of Computational and Graphical Statistics (JCGS)*, 16(1):44–66, 2007.