Provable guarantees for decision tree induction

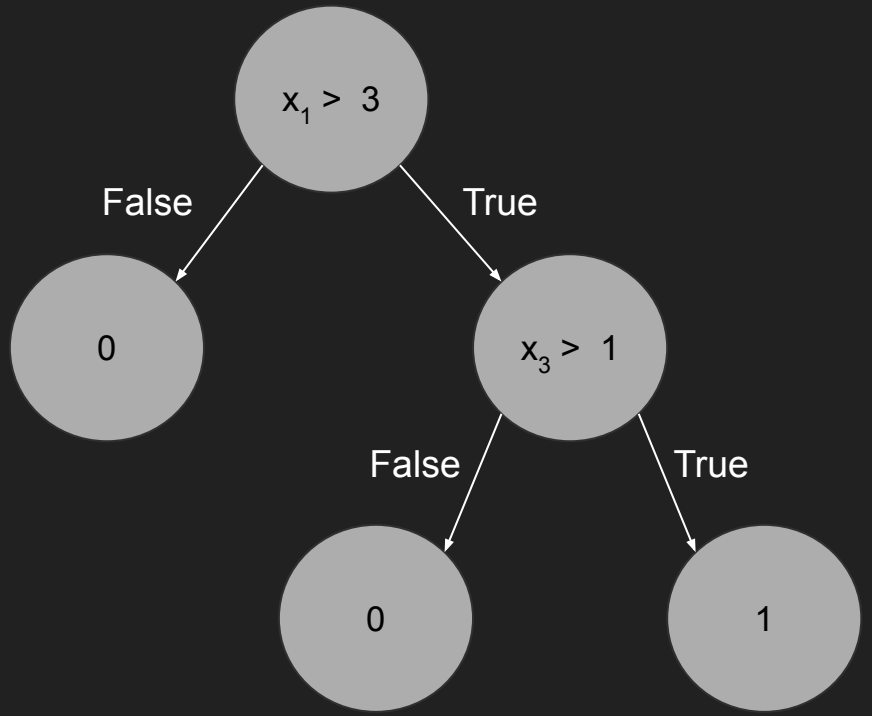
Guy Blanc*, Jane Lange*, Li-Yang Tan*


Stanford University

*Authors ordered alphabetically

Learning decision trees from labeled data

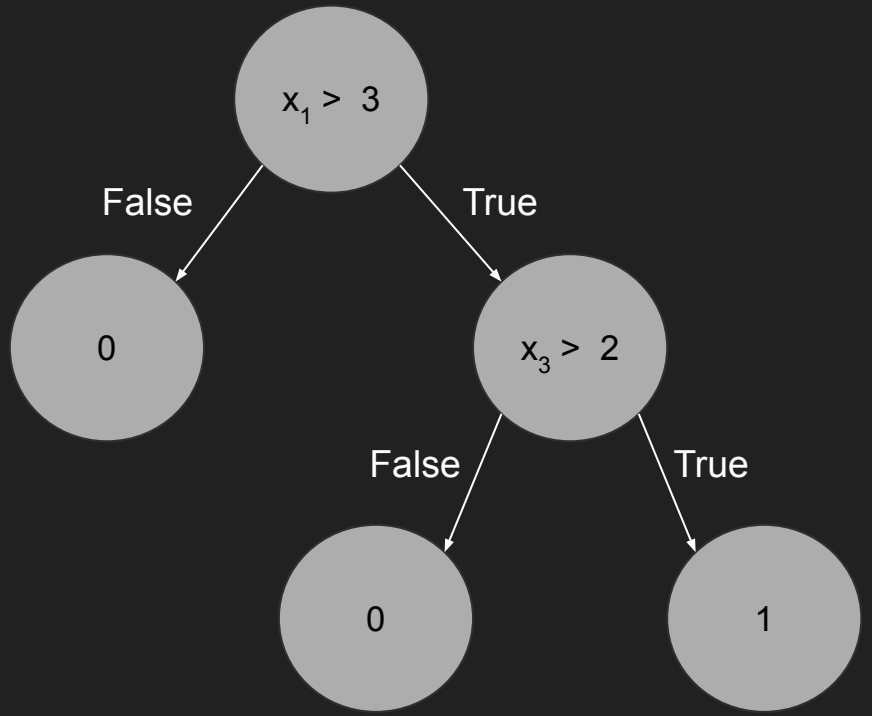
x	f(x)
[-3, 4.5, 8]	0
[5, 3, -1.2]	0
[4.2, -4, 5]	1
[4.1, 5.2, 2]	1
[-5, -10, 1.2]	0



Advantages of decision trees

Decision trees are:

- Interpretable
- Fast to evaluate
- Robust to scaling of features
- At the heart of modern ensemble methods like gradient boosted decision trees and random forests



Popular and empirically successful heuristics

- ID3: **Induction** of **decision trees** - Quinlan - Cited by 22574
- C4.5: **C4. 5**: **programs** for **machine learning** - Quinlan - Cited by 37670
- CART: **Classification** and **regression trees** - Breiman - Cited by 45727

Theory vs. practice of learning decision trees: A disconnect

Practical heuristics (ID3, C4.5, CART)

- Empirically successful and widely employed
- Lack theoretical guarantees

Learning Theory

- Decision trees one of the most intensively studied function classes [EH89, MR02]
- Algorithms developed do not resemble practical heuristics

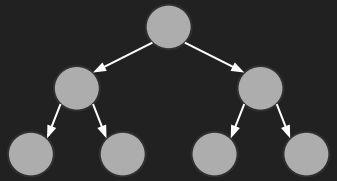
This talk: bridging this gap

This talk: Provable guarantees for DT heuristics

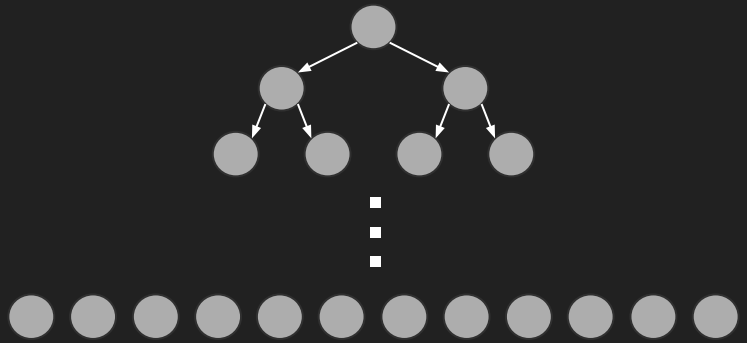
- Given a random sample from some function f
- Choose any size budget s
- The heuristics built a tree “not too much” larger than s, which achieves accuracy, relative to f, close to that of the best size s decision tree

Bad news

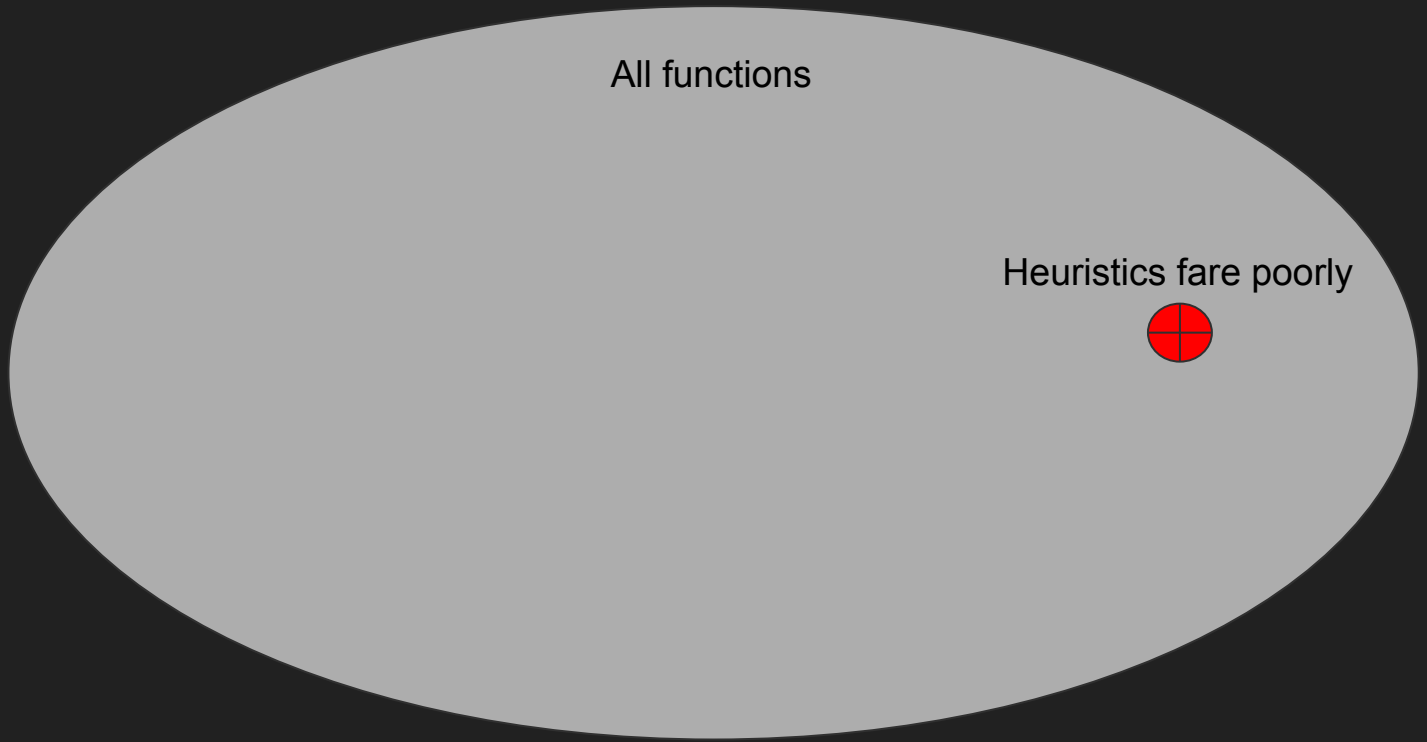
There are very simple f's for which all heuristics -- ID3, CART, C4.5 -- fare poorly. These $f : \mathbb{R}^d \rightarrow \{0,1\}$'s are constant size decision trees, but all heuristics build a decision tree of size 2^d



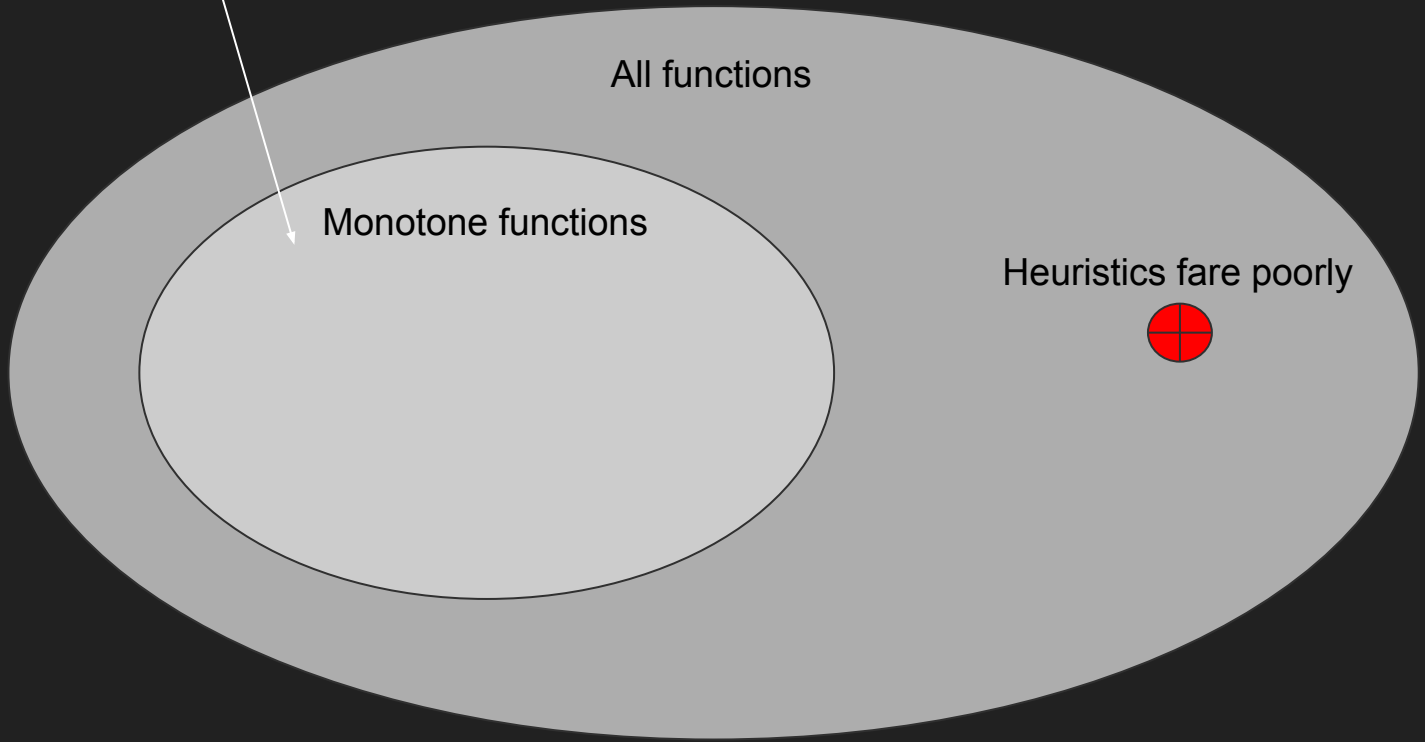
Exists decision tree with 4 leaves that computes f exactly



Heuristics build a decision tree with 2^d leaves


All functions
Heuristics fare poorly

Our provable guarantees



Monotonicity

We say that a function f is monotone if for all coordinates $i \in [d]$, f is non-decreasing in the i^{th} direction

Many real-world data sets are monotone

- Diagnostic criteria for a disease
- Evaluation criteria for a loan

Our results: Matching upper and lower bounds

Let $f : \mathbb{R}^d \to \{0,1\}$ be a monotone function and D a product distribution over $\mathbb{R}^d$

Let opt_s be the error of the best size-s decision tree for f, with respect to D

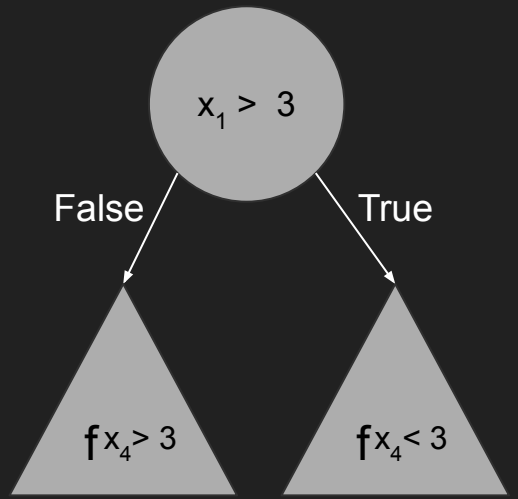
Upper bound: for all s and any error parameter ε, heuristics build tree of size $s^{O(\log(s)/\varepsilon^2)}$, with classification error $\text{opt}_s + \varepsilon$

Near-matching lower bound: there exists a monotone f and s such that $\text{opt}_s < 0.01$, yet top down heuristics cannot achieve error < 0.49 for any tree of size smaller than $s^{\Theta(\log s)}$

How do these heuristics work?
“Top-down induction of decision trees”

1) Determine “good” variable to query as root

2) Recurse on both subtrees



What is a “good” root?

Each heuristic uses an *impurity function* to assign a score to a candidate root

- ID3 and C4.5: binary entropy
- CART: Gini impurity

Impurity functions measure how far the leaves are from constant functions

Choose a root to maximize purity gain -- make the most progress toward constant leaves

Main technical ingredient in our analysis

Technique comes from analysis of boolean functions

OSSS inequality: every split makes good progress

- Every monotone decision tree has a highly-correlated variable
- Easy fact: thresholds with high correlation also have high purity gain

For the agnostic setting, need “robust” version of OSSS inequality

OSSS inequality + potential function argument: most leaves close to constant after few splits

We encode real-valued features as boolean vectors

Main result, summarized

If a monotone function is opt-close to a tree of size s:

Top-down heuristics learn a tree with error $\text{opt}_s + \varepsilon$

This tree has size $s^{O(\log s)}$

There are monotone functions for which this bound is tight

Further questions

Are there better guarantees for subclasses of monotone functions?

Are there better splitting criteria than these impurity functions?

What guarantees can be made for more advanced tree-based learning algorithms, such as boosted trees or random forests?