# Provable Guarantees for Decision Tree Induction


## Li-Yang Tan

Based on joint works with:

**Guy Blanc**
Stanford

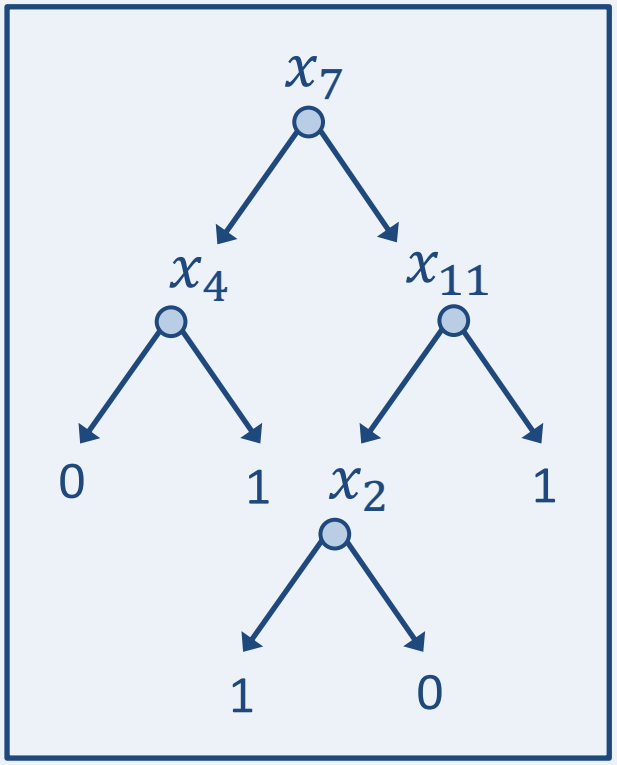
**Neha Gupta**
Stanford

**Jane Lange**
MIT

# Learning decision trees from labeled data

| Example $x \in \{0,1\}^n$ | Label $y \in \{0,1\}$ |
|---|---|
| 1 0 0 0 1 0 1 0 0 1 0 1 | 1 |
| 0 1 0 0 1 1 0 0 1 0 1 1 | 0 |
| 1 1 1 0 0 0 1 0 0 1 0 1 | 1 |
| 0 0 1 0 1 0 1 1 1 0 1 0 | 1 |
| 0 0 1 0 0 1 1 0 0 1 0 1 | 0 |
| 1 1 1 1 1 0 0 1 0 0 0 0 | 1 |
| 0 0 1 1 0 0 1 0 0 1 0 0 | 1 |
| 1 0 0 1 0 0 1 1 1 0 1 0 | 0 |
| 1 0 0 1 0 1 0 1 0 1 1 0 | 1 |

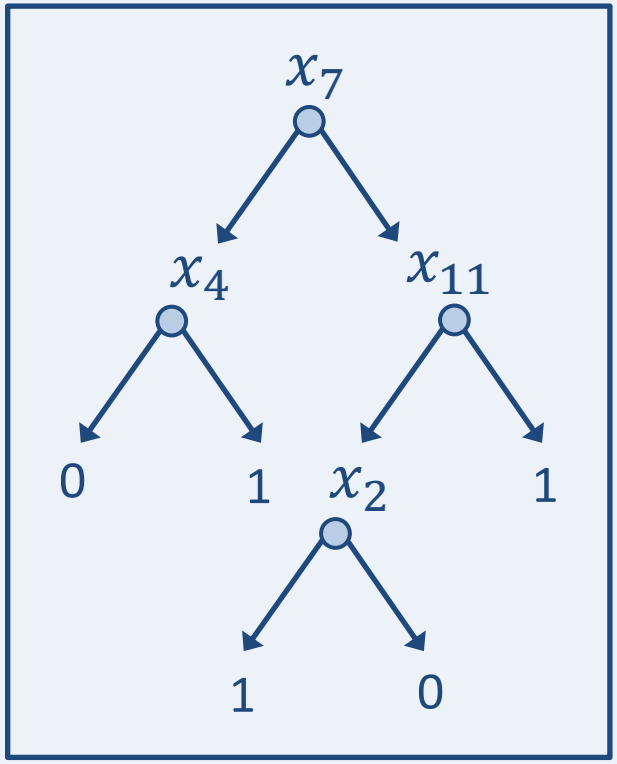
**Fast** algorithm




**Small** decision tree

# Decision trees: simple and effective

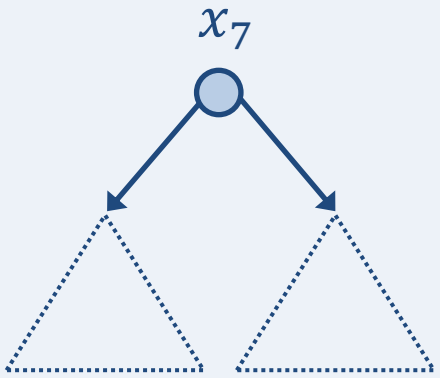
- Fast to evaluate
- Easy to understand,
  Easy to explain predictions
- Powerful “ensemble methods”:
  Random forests and Boosted decision trees



# Top-down algorithms for learning DTs

**A natural approach:** Greedy, top-down

1. Choose a "good" variable to query as the root
2. Recurse



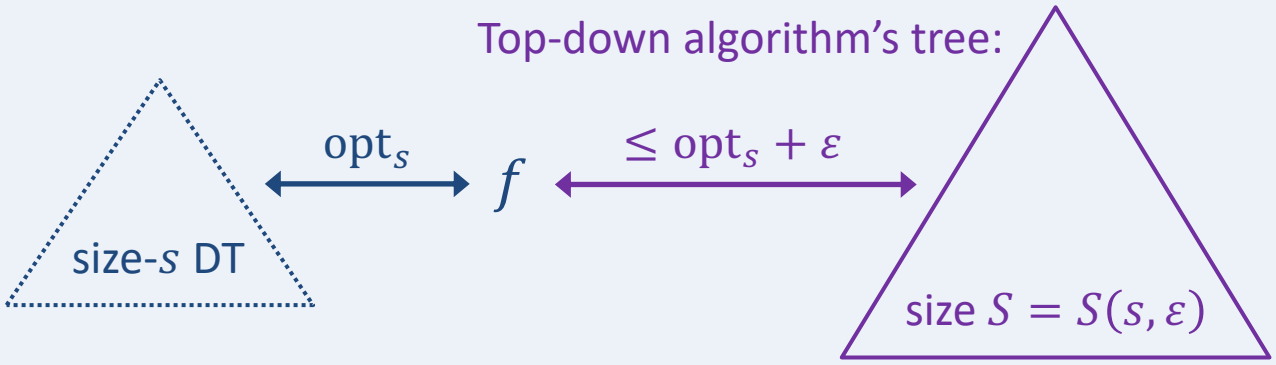

- Definition of "good" = splitting criterion of algorithm
  - Intuitively, want the most "important"/"relevant" variable
- Captures almost all algorithms used in practice: ID3, C4.5, CART, etc.
  - Widely employed, empirically successful
  - **This talk:** theoretical justification for their effectiveness?

# This talk: Performance guarantees for top-down algorithms

Let $f:\{0,1\}^n \to \{0,1\}$ be a target function, $\mathcal{D}$ a distribution over $\{0,1\}^n$

**Input:** Labeled examples $\big(x, f(x)\big)$ where $x \sim \mathcal{D}$,
Parameters $s \in \mathbb{N}$, and $\varepsilon \in (0,1)$

**Goal:** Top-down algorithms achieve error within $\varepsilon$ of that of the optimal size-$s$ tree for $f$, with a tree "not much larger than" $s$

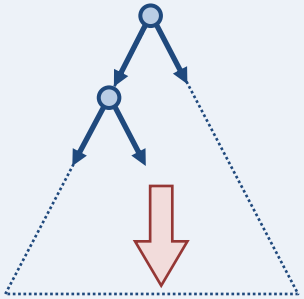



**Our results:** $\mathcal{D}$ = product distribution (This talk: $\mathcal{D}$ = uniform)

# Learning decision trees: practice vs. theory

Decision trees long been a workhorse of ML

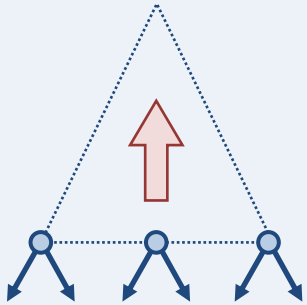
- Algorithms used in practice proceed in top-down fashion:



- Empirically successful, but lack performance guarantees

Rich and vast theory literature on learning decision trees

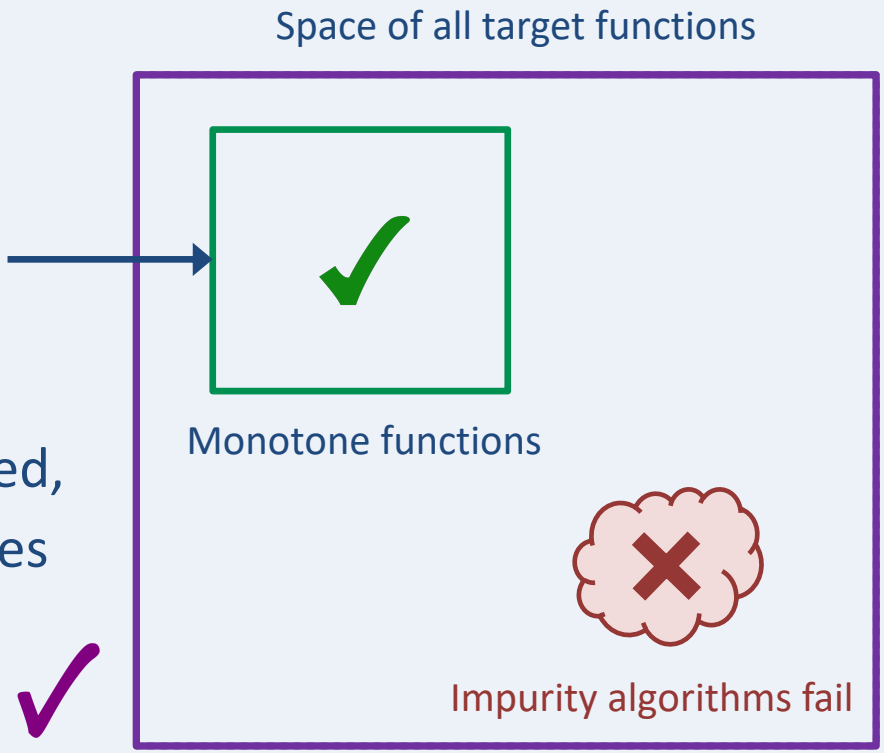
- Many algorithms do not return a decision tree hypothesis ("improper" algorithms)
- Fastest proper algorithm [Ehrenfeucht–Haussler 89] proceeds *bottom-up*



# Outline of this talk

- How these top-down algorithms work: “impurity-based”
- **Bad news:** target functions for which impurity algorithms fail badly

- **Result I:** Performance guarantees for monotone functions
- **Result II:** A new, non-impurity-based, algorithm which achieves guarantees for all functions



# Part I:

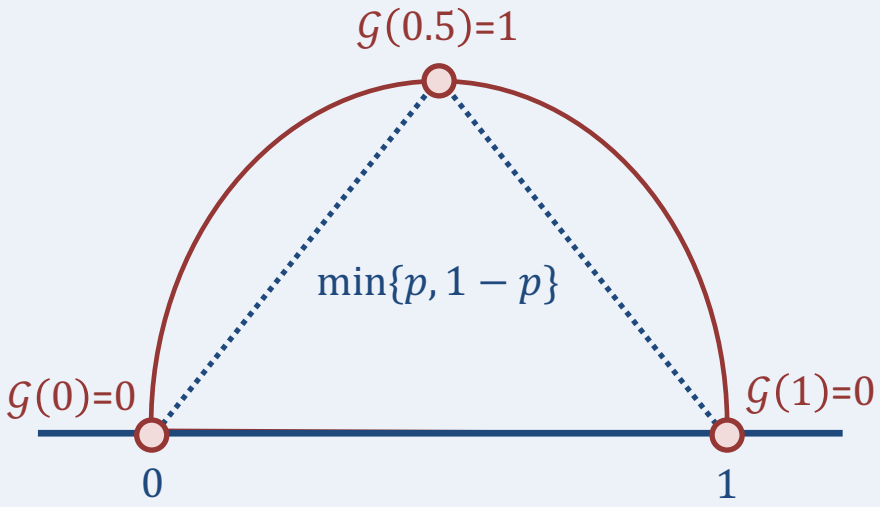
# Power and limitations of existing algorithms

"Top-down induction of decision trees: rigorous guarantees and inherent limitations",
Guy Blanc, Jane Lange, Li-Yang Tan. ITCS 2020.

"Provable guarantees for decision tree induction: the agnostic setting",
Guy Blanc, Jane Lange, Li-Yang Tan. ICML 2020.

# How do these top-down algorithms work?

- Main algorithms used in practice – ID3, C4.5, CART – all "impurity-based"
- Each defined by an impurity function $\mathcal{G}: [0,1] \to [0,1]$
  - Concave, satisfies $\mathcal{G}(0) = \mathcal{G}(1) = 0$ and $\mathcal{G}(0.5) = 1$




Examples:

ID3 uses $\mathcal{G}(p) = \mathrm{H}_2(p)$, binary entropy

CART uses $\mathcal{G}(p) = 4p(1-p)$, "Gini impurity"

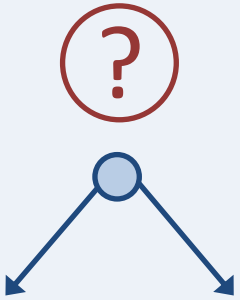
**High-level idea**

$\mathcal{G}(p)$ approximation of $\min\{p, 1-p\}$

$\therefore$ For $f: \{0,1\}^n \to \{0,1\}$,

$$\mathcal{G}(\mathbb{E}[f]) \approx \underbrace{\min\{\mathbb{E}[f], 1-\mathbb{E}[f]\}}_{\text{How close } f \text{ is to a constant function}},$$

# Impurity function defines the splitting criterion

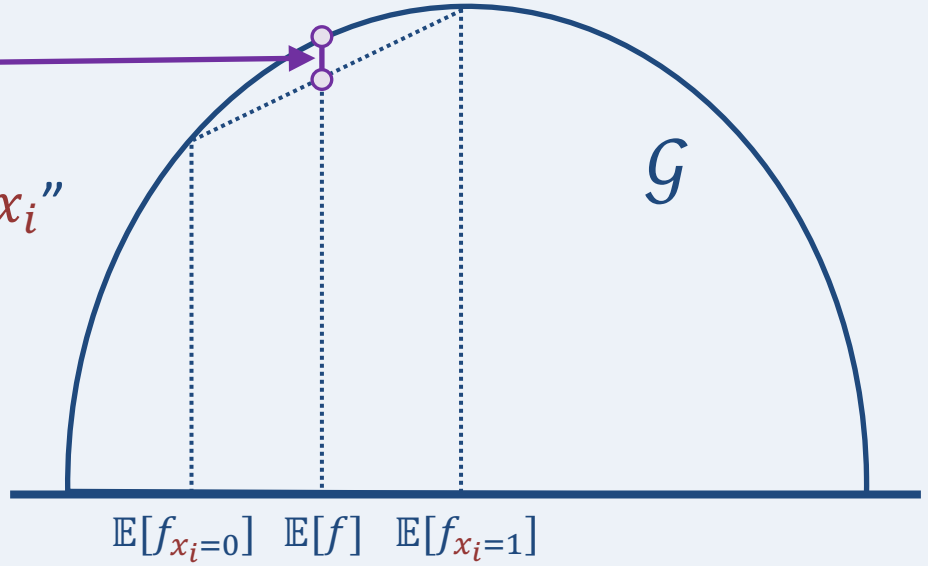



Target function $f: \{0,1\}^n \to \{0,1\}$. Which $x_i$ to query as the root?

Query $x_i$ as the root, where $i \in [n]$ maximizes:

$$\mathcal{G}(\mathbb{E}[f]) - \mathbb{E}_b\left[\mathcal{G}\left(\mathbb{E}\left[f_{x_i=b}\right]\right)\right]$$

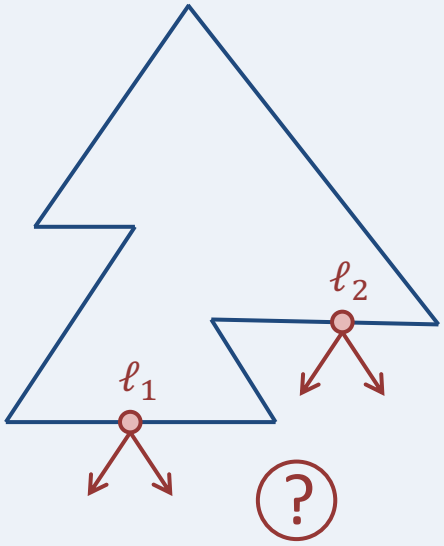
"Purity gain associated with splitting $x_i$"



# Order in which to split the leaves

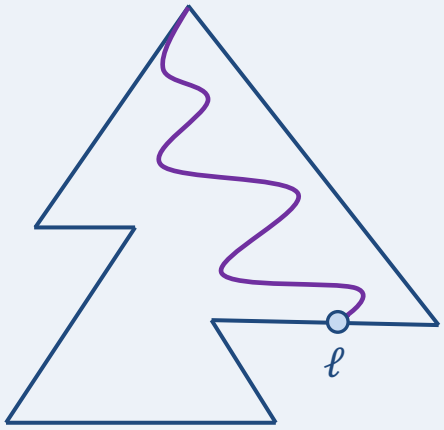
Here's the tree we've built so far:

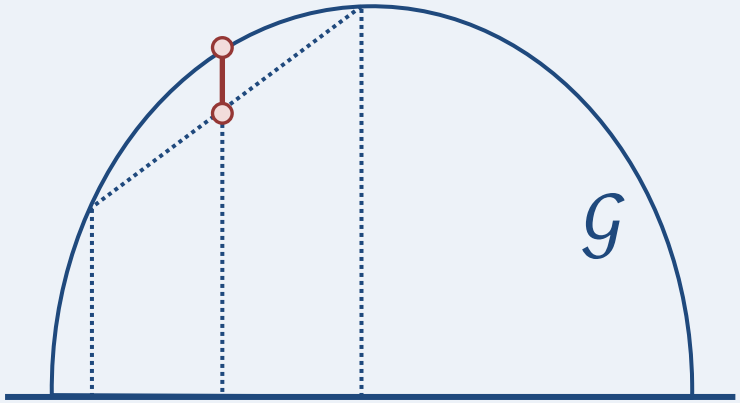
Which leaf to split next?




We split the leaf $\ell$ with the highest "**score**", where:

$$\text{score}(\ell) := \mathbb{P}_{x\sim\mathcal{D}}[x \text{ reaches } \ell] \cdot \max_{i\in[n]} \{ \text{Purity-gain}_{\mathcal{G}}(\ell, x_i) \}$$



# A single representative algorithm?

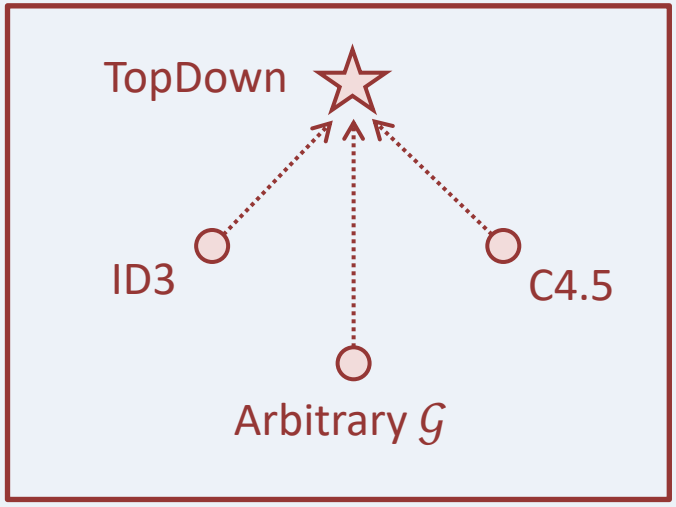
Different impurity functions $\mathcal{G}$ give rise to different trees.

Would be nice to analyze a **single** algorithm …

**Lemma** [Blanc-Lange-T.]**:**

There is an algorithm, **TopDown**, that is "complete" for impurity algorithms:

Guarantees for TopDown yield guarantees for all impurity algorithms



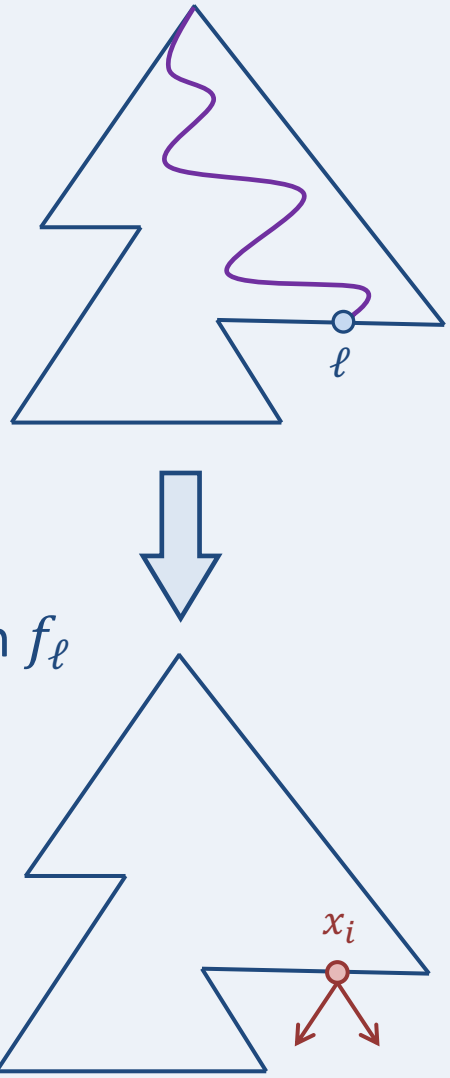

Overhead in our reduction scales with concavity of $\mathcal{G}$

# TopDown

Forget about impurity functions. In each iteration, we split the leaf $\ell$ with the highest **score**, where:

$$\text{score}(\ell) := \mathbb{P}_{x\sim\mathcal{D}}[x \text{ reaches } \ell] \cdot \max_{i\in[n]} \{\underbrace{\mathbb{E}[f_\ell(x)x_i]^2}\}$$

"correlation btw $f_\ell$ and $x_i$"




Split $\ell$ with a query to the $x_i$ that's best correlated with $f_\ell$

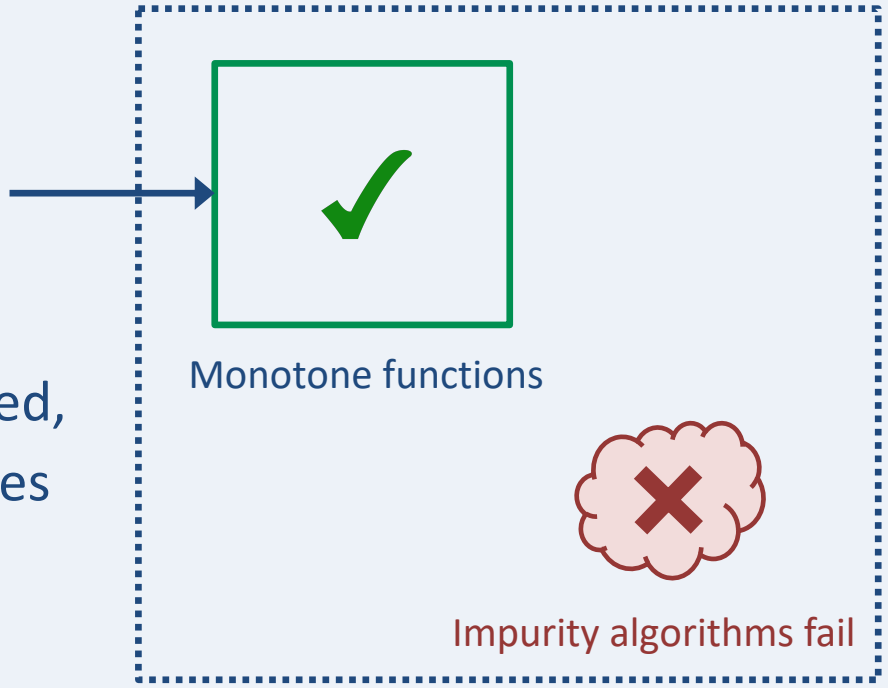
**How does the process end?**

When tree reaches certain user-specified size $S$, label each leaf $\ell$ with $\text{round}(\mathbb{E}[f_\ell])$ and output tree

# Outline of this talk

- How these top-down algorithms work: "impurity-based"
- **Bad news:** target functions for which impurity algorithms fail badly

- **Result I:** Performance guarantees for monotone functions
- **Result II:** A new, non-impurity-based, algorithm which achieves guarantees for all functions



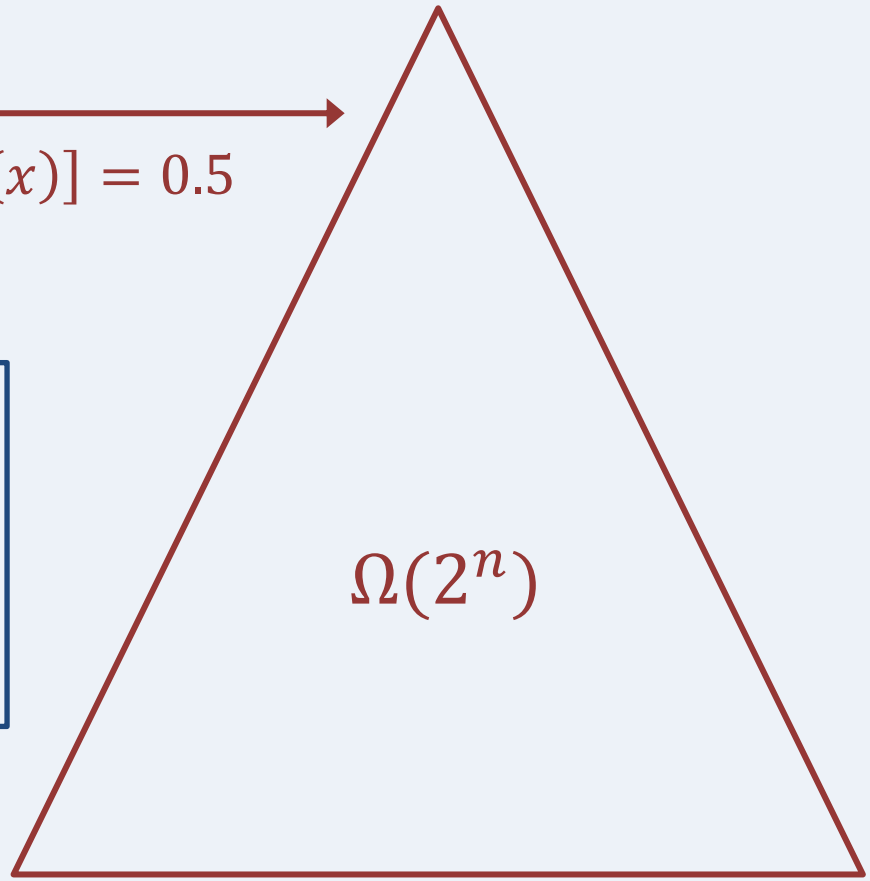
# Bad news for TopDown

There is an $f: \{0,1\}^n \to \{0,1\}$ such that:

$\equiv \quad f$

$\mathbb{P}[T(x) \neq f(x)] = 0.5$

TopDown's tree $T$




$f(x) = x_i \oplus x_j$ for some $i, j \in [n]$

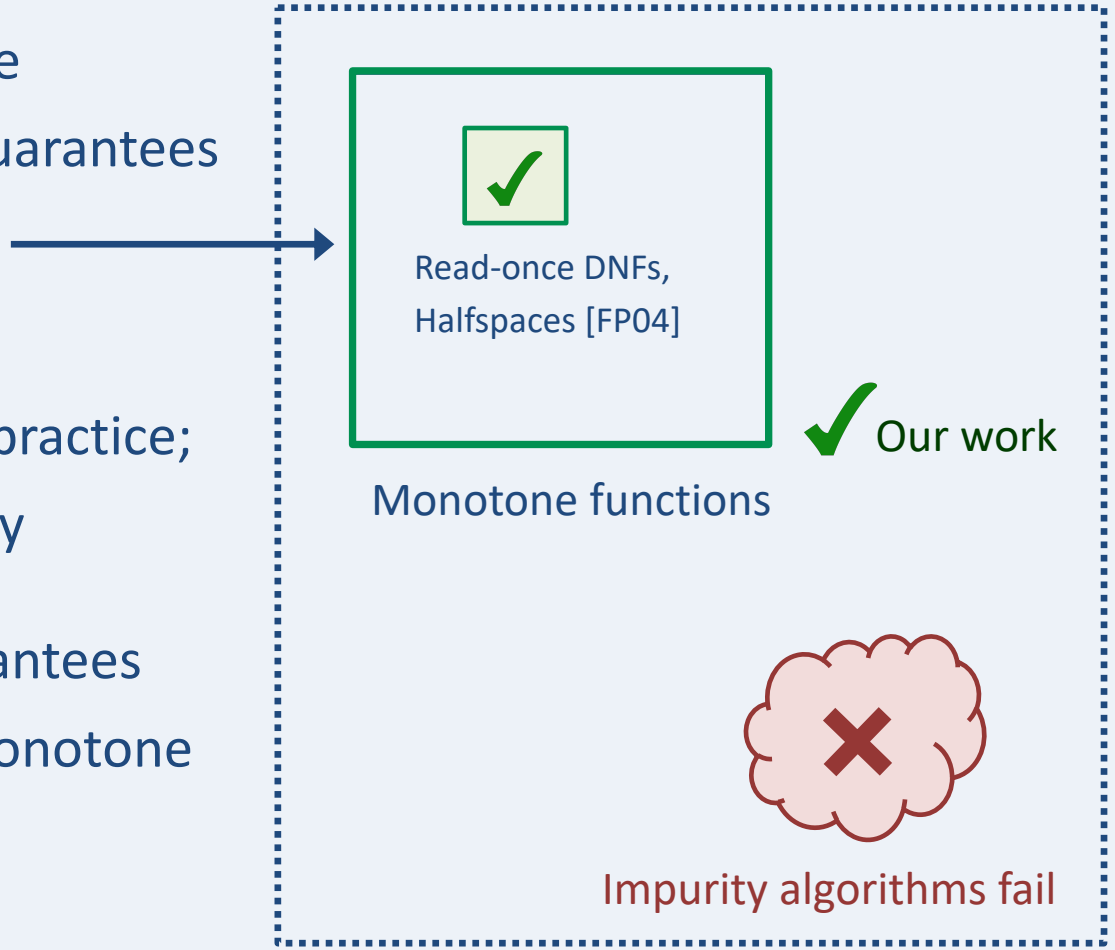
TopDown struggles because:

$\mathbb{E}[f(x)x_k] = 0$ for all $k \in [n]$

M. Kearns, "Recent developments in decision tree induction and the weak learning framework", AAAI 1996.

# Guarantees for monotone functions?

[Fiat–Pechyony 04] raised the possibility of performance guarantees for **monotone** functions




- Reasonable assumption in practice; intensively studied in theory
- [FP04]: Proved strong guarantees for various subclasses of monotone functions

**Theorem** [Blanc-Lange-T.]**:**

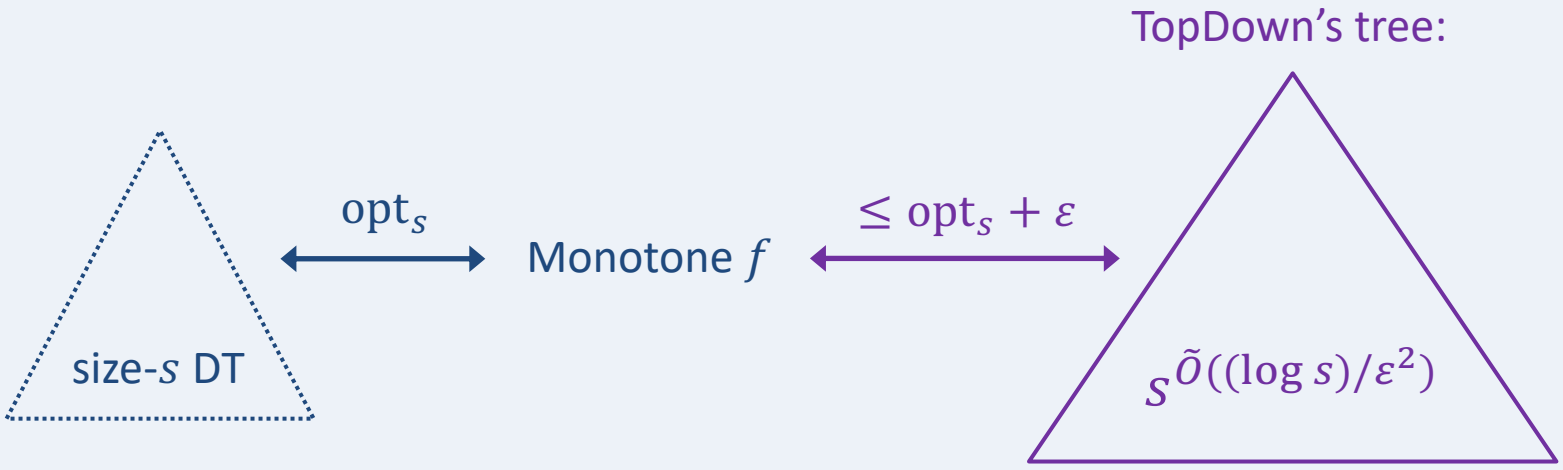
Let $f:\{0,1\}^n \to \{0,1\}$ be monotone, $\mathcal{D}$ = product distribution over $\{0,1\}^n$.

For all $s \in \mathbb{N}$ and $\varepsilon \in (0,1)$, given examples $\big(x, f(x)\big)$ where $x \sim \mathcal{D}$,

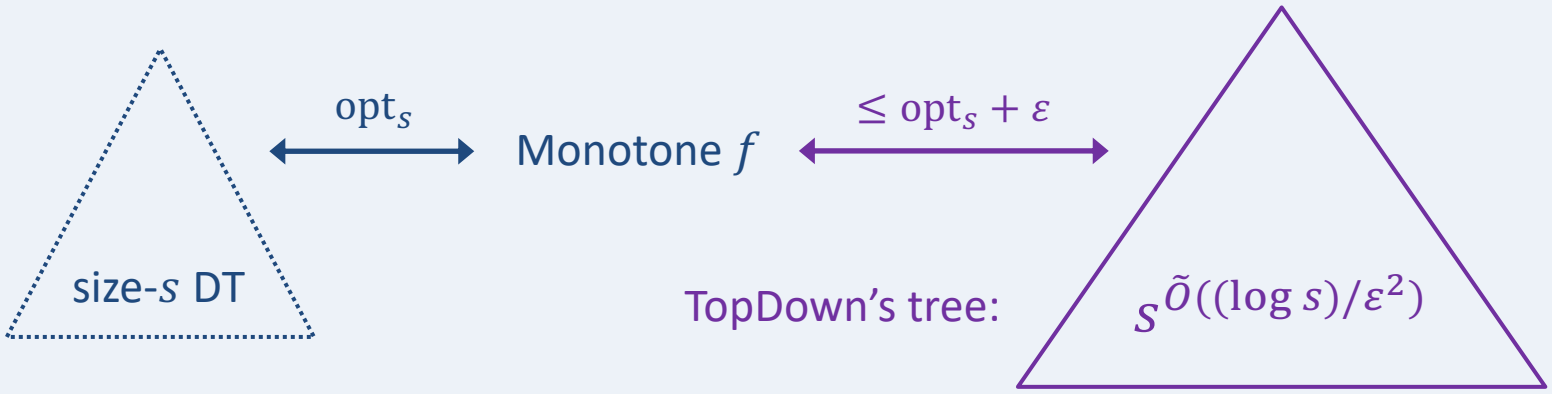
TopDown grows a tree $T$ of size

$$s^{\tilde{O}((\log s)/\varepsilon^2)}$$

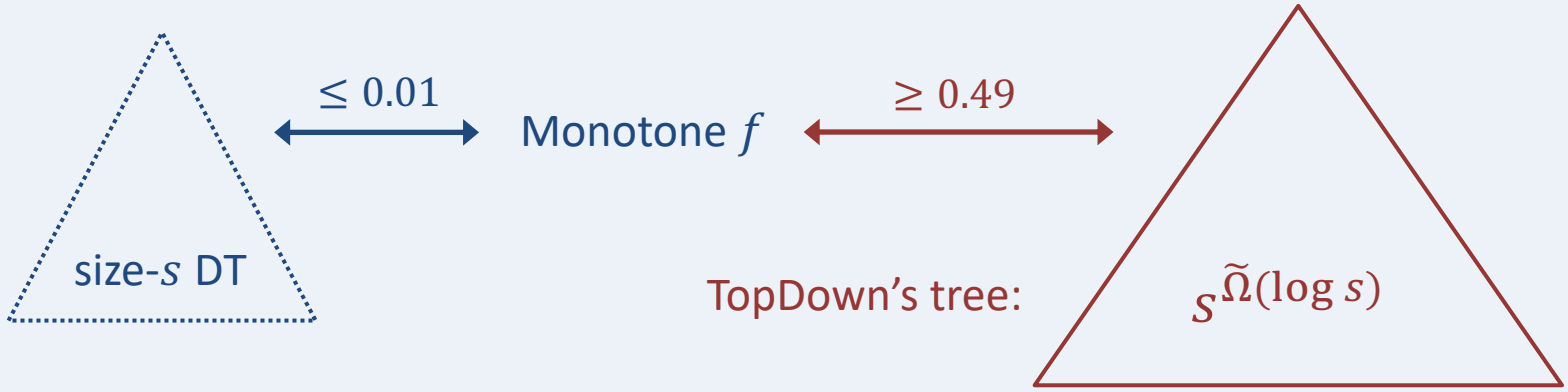
that achieves $\mathbb{P}_{x\sim\mathcal{D}}[f(x) \neq T(x)] \leq \mathrm{opt}_s + \varepsilon$.



**Theorem:** For all $s \in \mathbb{N}$, $\varepsilon \in (0,1)$, and monotone $f$,

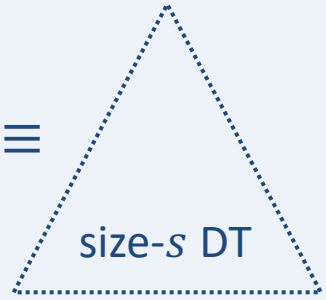



**Matching lower bound:** For all $s \in \mathbb{N}$, there is a monotone $f$ such that:

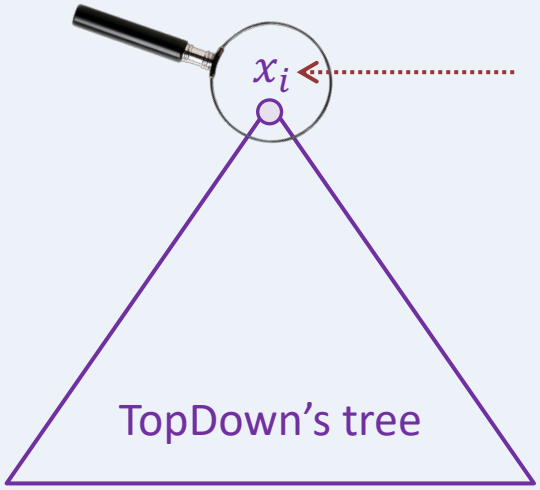


# Proof sketch of the upper bound

Consider realizable setting ($\mathrm{opt}_s = 0$)




Let's focus on understanding the **first** decision that TopDown makes:




**Q:** What can we say about $\mathbb{E}[f(x)x_i]$?

What's the relationship between

$\max_{i \in [n]} \mathbb{E}[f(x)x_i]$ and the complexity of $f$?

# O'Donnell, Saks, Schramm, Servedio

"Every decision tree has an influential variable", FOCS 2005

**OSSS inequality:** If $f:\{0,1\}^n \to \{0,1\}$ is computed by a size-$s$ decision tree, there is an $i \in [n]$ such that:

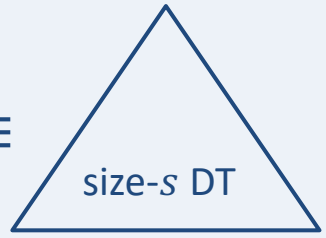
$$\mathrm{Inf}_i(f) \geq \frac{\mathrm{Var}(f)}{\log s}$$

where $\mathrm{Inf}_i(f) := \mathbb{P}\left[f(x) \neq f(x^i)\right]$ and $x^i = x$ with $i$-th bit flipped.

- If $f$ is monotone, $\mathrm{Inf}_i(f) = \mathbb{E}[f(x)x_i]$
- If $\mathrm{Var}(f) \leq \varepsilon$ then $f$ is $O(\varepsilon)$-close to constant

# OSSS vs. Kahn-Kalai-Linial inequality

**OSSS inequality**

$f \equiv$ 


Exists $i \in [n]$ such that:

$$\mathrm{Inf}_i(f) \geq \frac{\mathrm{Var}(f)}{\log s}$$

**KKL inequality**

For all $f: \{0,1\}^n \to \{0,1\}$

Exists $i \in [n]$ such that:

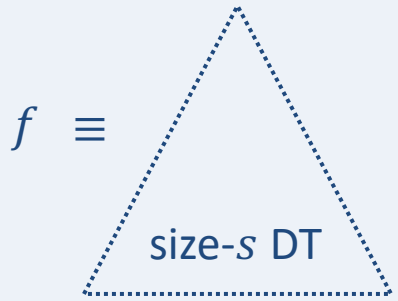
$$\mathrm{Inf}_i(f) \geq \mathrm{Var}(f) \cdot \frac{\log n}{n}$$

- OSSS = "KKL that takes into account decision tree complexity of $f$"
- Strengthens KKL whenever $s = 2^{o(n/\log n)}$
  - Exponentially stronger when $s = \mathrm{poly}(n)$

# Back to analyzing TopDown

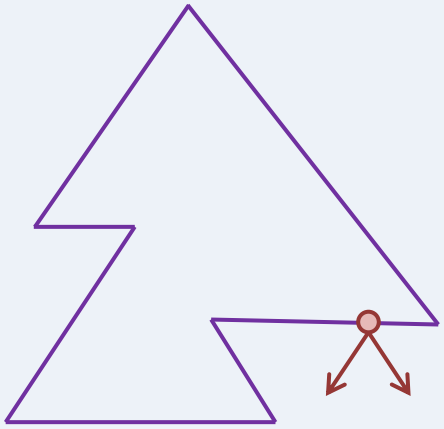
**Simplifying assumption #1**

Realizable setting ($\text{opt} = 0$)



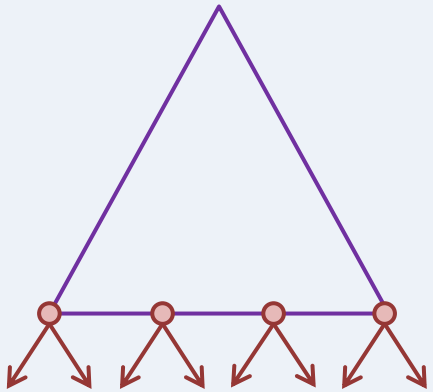

**Simplifying assumption #2**

How TopDown works:



Splits leaf with highest score

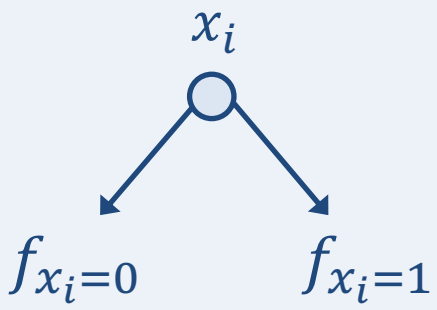
What we'll instead analyze:



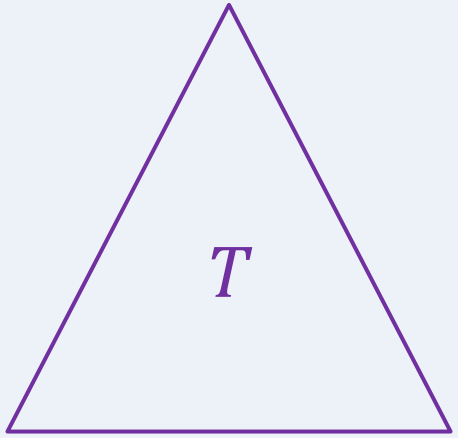
Splits **all** leaves,
maintains complete tree

# A simple potential function argument

How does $\mathrm{Var}(f)$ change when $x_i$ is queried?




**Fact:** $\mathbb{E}_b\left[\mathrm{Var}\left(f_{x_i=b}\right)\right] = \mathrm{Var}(f) - \mathbb{E}_x[f(x)x_i]^2$

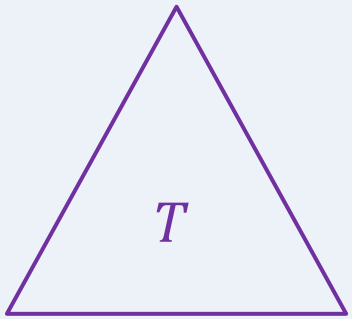



TopDown's tree so far

Consider the quantity: $\mathbb{E}_{\substack{\text{leaves} \\ \ell \sim T}}[\mathrm{Var}(f_\ell)]$

- When $T = \emptyset$, this is simply $\mathrm{Var}(f)$
- For all $T$, this quantity is $\geq 0$

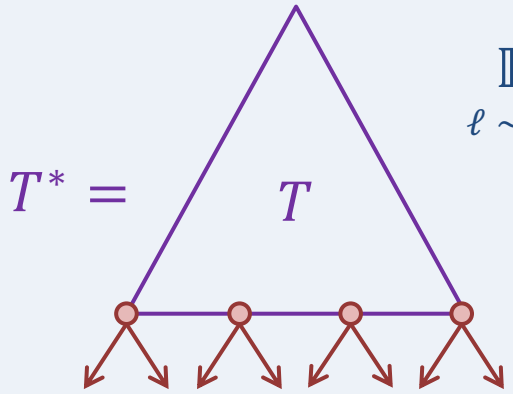
TopDown's tree so far




**Q:** Is it true that for $\geq 1-\varepsilon$ fraction of leaves $\ell$, $\mathrm{Var}(f_\ell) \leq \varepsilon$?

**If so,** done: $T$ is $O(\varepsilon)$-close to $f$

**If not,** split all leaves of $T$:




Variable maximizing correlation with $f_\ell$

$$\mathop{\mathbb{E}}_{\ell \sim T^*}[\mathrm{Var}(f_\ell)] = \mathop{\mathbb{E}}_{\ell \sim T}[\mathrm{Var}(f_\ell)] - \mathop{\mathbb{E}}_{\ell \sim T}\left[\underbrace{\mathbb{E}_x[f_\ell(x)x_{i(\ell)}]^2}\right]$$

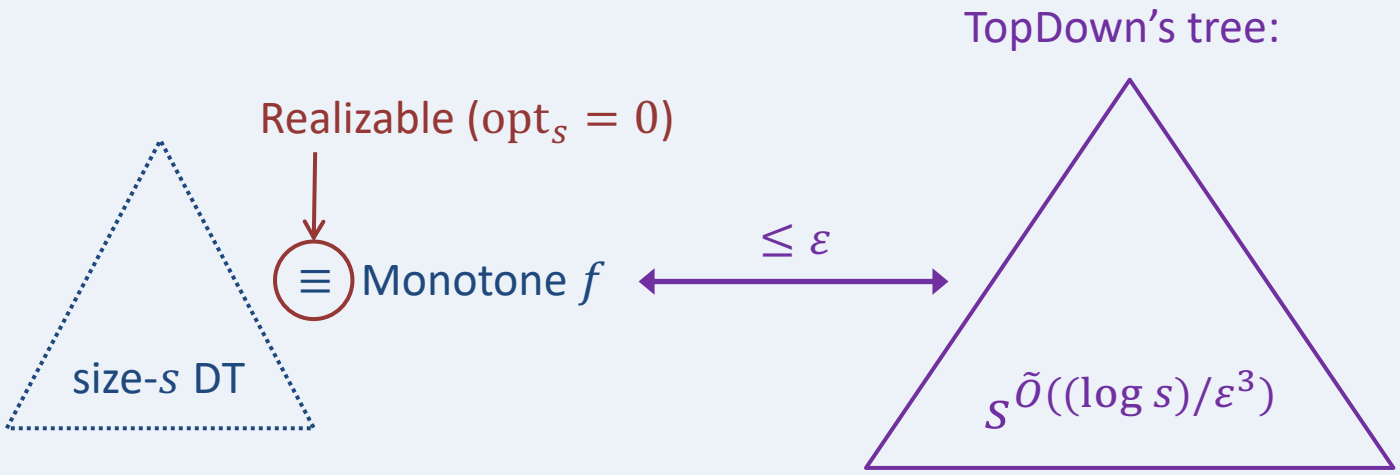
$\geq \varepsilon$ fraction s.t. $\mathrm{Var}(f_\ell) \geq \varepsilon$ $\qquad \geq \left(\dfrac{\mathrm{Var}(f_\ell)}{\log s}\right)^2$ [OSSS]

1 unit of depth buys us $\geq \varepsilon^3/(\log s)^2$ decrease in "potential"

$\therefore$ Done when depth $\leq (\log s)^2/\varepsilon^3$, hence size $\leq s^{\tilde{O}((\log s)/\varepsilon^3)}$. $\square$

# Beyond the proof sketch

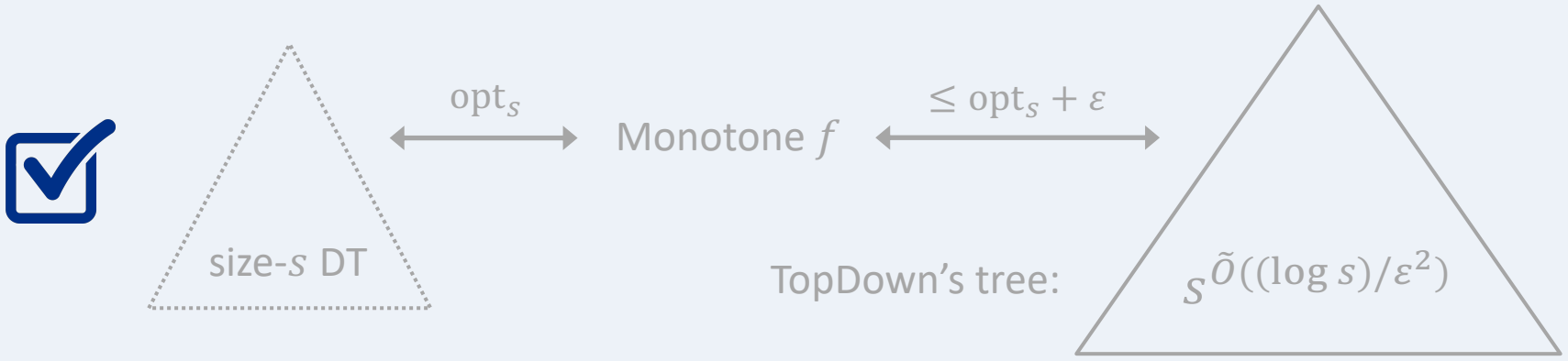
We just sketched the proof of:




- For **non-realizable** setting, need "robust" version of OSSS inequality, proved by [Jain-Zhang 11].

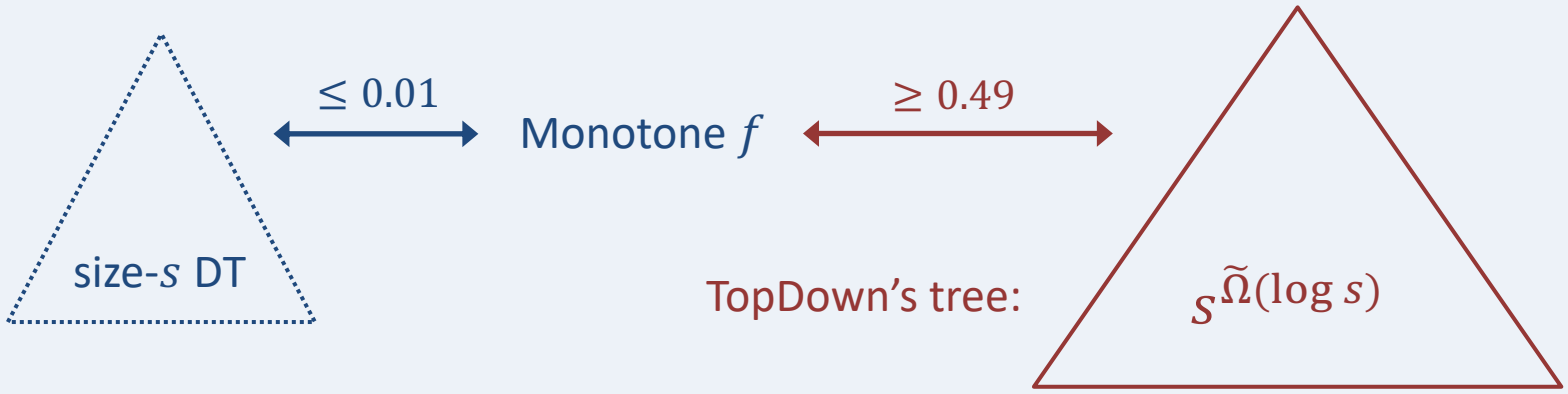
- For **realizable** setting, can optimize argument to get slightly better parameters: $s^{O((\log s)^{1/2}/\varepsilon)}$ instead of $s^{\tilde{O}((\log s)/\varepsilon^3)}$.

Theorem: For all $s \in \mathbb{N}$, $\varepsilon \in (0,1)$, and monotone $f$,




**Next:**

**Matching lower bound:** For all $s \in \mathbb{N}$, there is a monotone $f$ such that:



# A monotone function that fools TopDown

Our function will be based on Tribes and Majority:

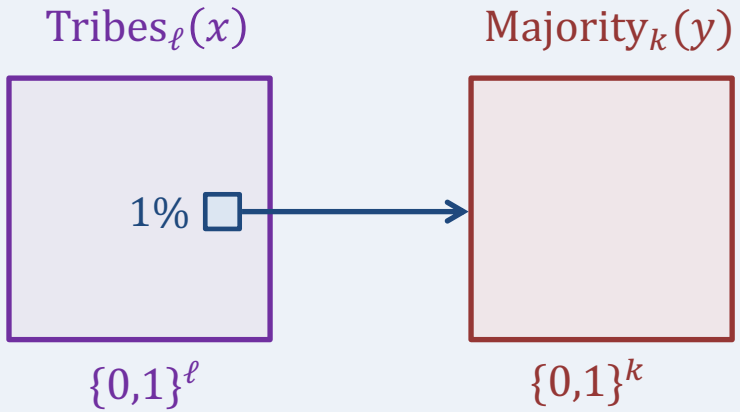
$\text{Tribes}_\ell(x_1, \dots, x_\ell)$: $\mathbb{E}[\text{Tribes}_\ell(x)x_i] = \Theta\left(\frac{\log \ell}{\ell}\right)$ for all $i \in [\ell]$

$\text{Majority}_k(y_1, \dots, y_k)$: $\mathbb{E}[\text{Majority}_k(y)y_i] = \Theta\left(\frac{1}{\sqrt{k}}\right)$ for all $i \in [k]$

**High-level idea:** Our function $F(x_1, \dots, x_\ell, y_1, \dots, y_k)$ will be:

- "99% $\text{Tribes}_\ell$ + 1% $\text{Majority}_k$", so $F \approx \text{Tribes}_\ell$
- However, $\ell$ and $k$ chosen so that $\text{Majority}_k$'s variables better correlated with $F$ than $\text{Tribes}_\ell$'s

  $\therefore$ TopDown fooled into querying $\text{Majority}_k$'s variables




$F(x_1, \dots, x_\ell, y_1, \dots, y_k) :=$

If $x \notin \square$ output $\text{Tribes}_\ell(x)$

Else output $\text{Majority}_k(y)$

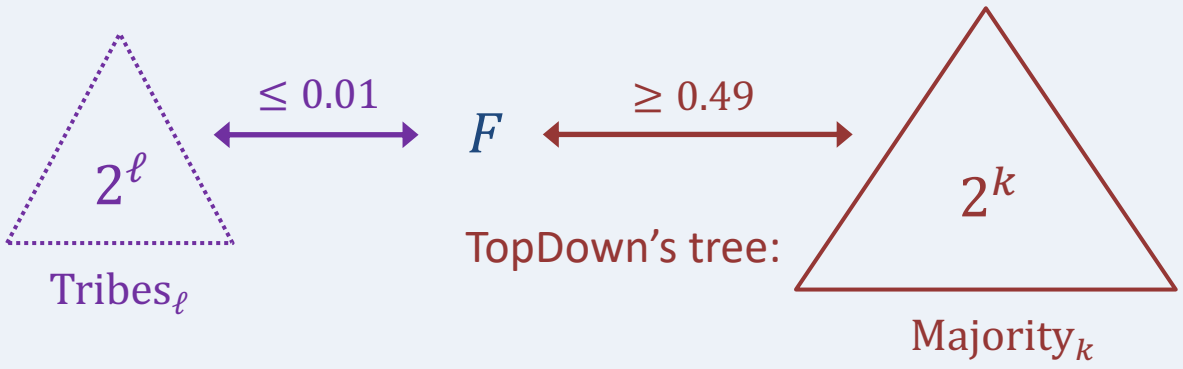
---

**Observation:** $F$ is 0.01-close to $\text{Tribes}_\ell(x)$, decision tree size $\leq 2^\ell$.

We choose $\ell$ and $k$ to satisfy $\frac{1}{\sqrt{k}} \gtrsim \frac{\log \ell}{\ell}$,

ensuring that $\mathbb{E}[F(x,y)y_i] \gtrsim \mathbb{E}[F(x,y)x_i]$.

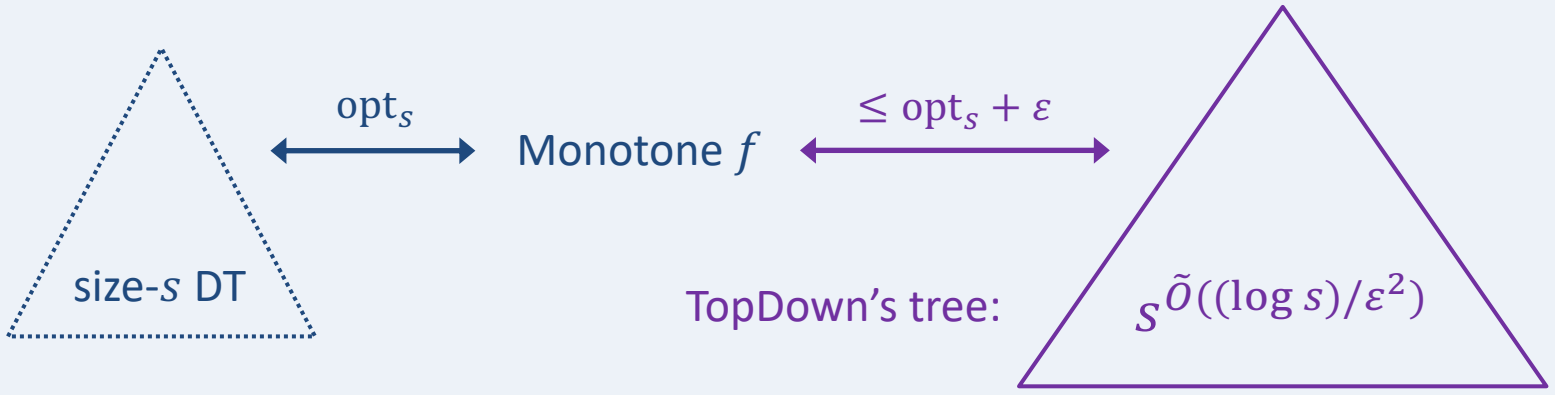
$\therefore$ TopDown fooled into building DT for $\text{Majority}_k(y)$, size $\geq 2^k$.



# End of Part I

Performance guarantees for impurity algorithms when run on monotone target functions:

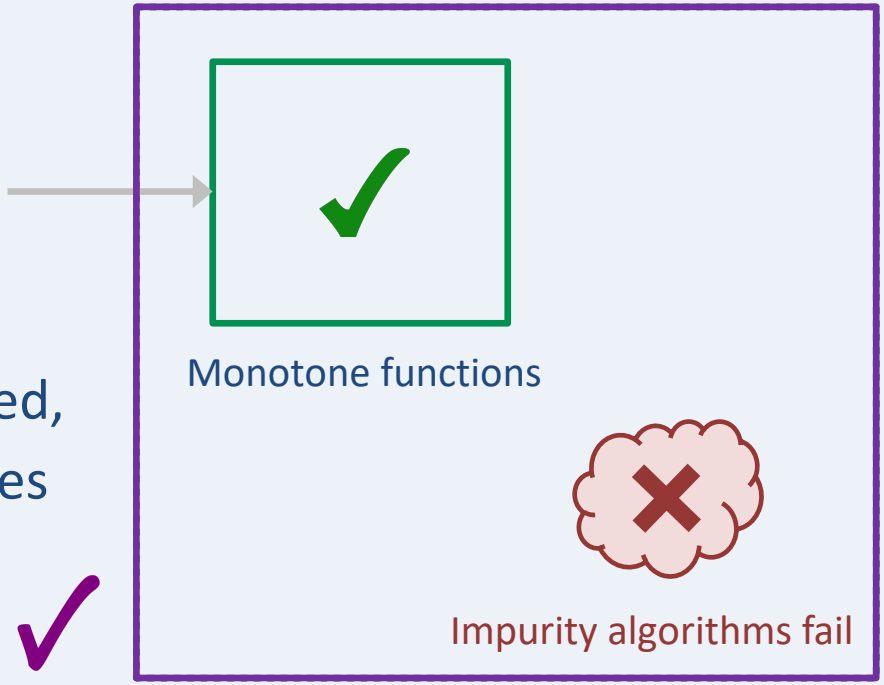
**Theorem:** For all $s \in \mathbb{N}$, $\varepsilon \in (0,1)$, and monotone $f$,




We provide a matching $s^{\tilde{\Omega}(\log s)}$ lower bound

# Outline of this talk

- How these top-down algorithms work: "impurity-based"
- **Bad news:** target functions for which impurity algorithms fail badly
- **Result I:** Performance guarantees for monotone functions
- **Result II:** A new, non-impurity-based, algorithm which achieves guarantees for all functions



# Part II:

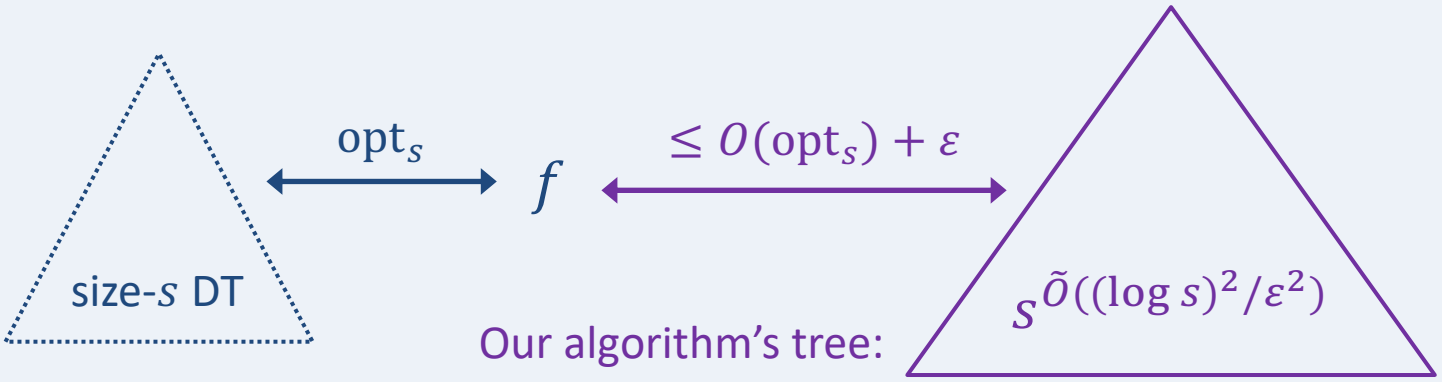
## A new splitting criterion

"Universal guarantees for decision tree induction via a higher-order splitting criterion", Guy Blanc, Neha Gupta, Jane Lange, Li-Yang Tan. NeurIPS 2020.

# Our main result

A new splitting criterion and associated learning algorithm that achieves performance guarantees for **all** functions:

**Theorem:** For all $s \in \mathbb{N}$, $\varepsilon \in (0,1)$, and all target functions $f$,



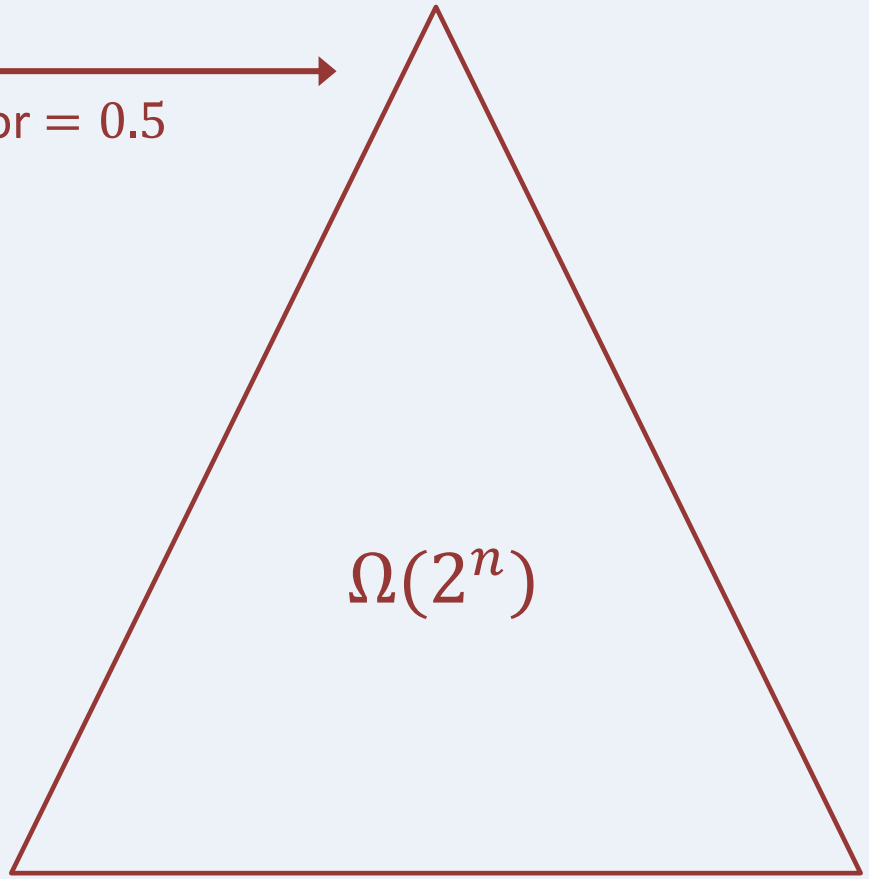

Our algorithm can be viewed as a "noise stabilization" procedure

# Revisiting the bad news for TopDown

$f(x) = x_i \oplus x_j$

TopDown's tree $T$

error $= 0.5$



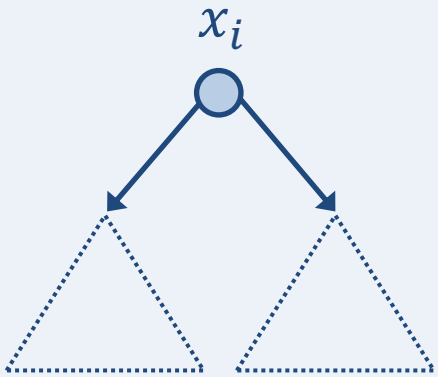

TopDown struggles because:

$\mathbb{E}[f(x)x_k] = 0$ for all $k \in [n]$

**Easy fix for this example:**

Take pairwise correlations $\mathbb{E}[f(x)x_k x_\ell]$ into account

# A generalized splitting criterion




**TopDown's splitting criterion**

Split $x_i$ where $i \in [n]$ maximizes $\mathbb{E}[f(x)x_i]^2$

**Generalized splitting criterion**

Split $x_i$ where $i \in [n]$ maximizes $\sum_{\substack{S \ni i \\ |S| \le d}} (1-\delta)^{|S|} \cdot \mathbb{E}\left[f(x)\prod_{j \in S} x_j\right]^2$

Sets of size $\le d$ containing $i$

Correlation between $f$ and variables in $S$

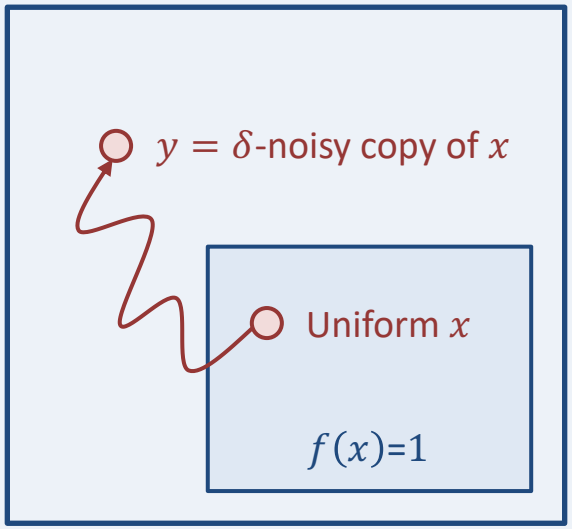
"Attenuating" factor $(1-\delta)^{|S|}$: large sets contribute less (necessary!)

# Combinatorial view: “Noisy influence”

Split $x_i$ where $i \in [n]$ maximizes $\sum_{\substack{S \ni i \\ |S| \le d}} (1-\delta)^{|S|} \cdot \mathbb{E}\left[f(x) \prod_{j \in S} x_j\right]^2$

**Def.** Noisy sensitivity of $f$ at rate $\delta$

$$\mathrm{NS}_\delta[f] \coloneqq \mathbb{P}_{y \underset{\delta}{\sim} x}[f(x) \neq f(y)]$$




$\asymp$ Expected drop in noise sensitivity from querying $x_i$

$$\mathrm{NS}_\delta(f) - \mathbb{E}_{b \sim \{0,1\}}\left[\mathrm{NS}_\delta\left(f_{x_i=b}\right)\right]$$

# Our algorithm vs. TopDown

- Recall simple key fact in our analysis of TopDown:

$$\mathbb{E}_x[f(x)x_i]^2 = \mathrm{Var}(f) - \mathbb{E}_b\left[\mathrm{Var}\left(f_{x_i=b}\right)\right]$$

TopDown greedily drives down the variance of $f$

---

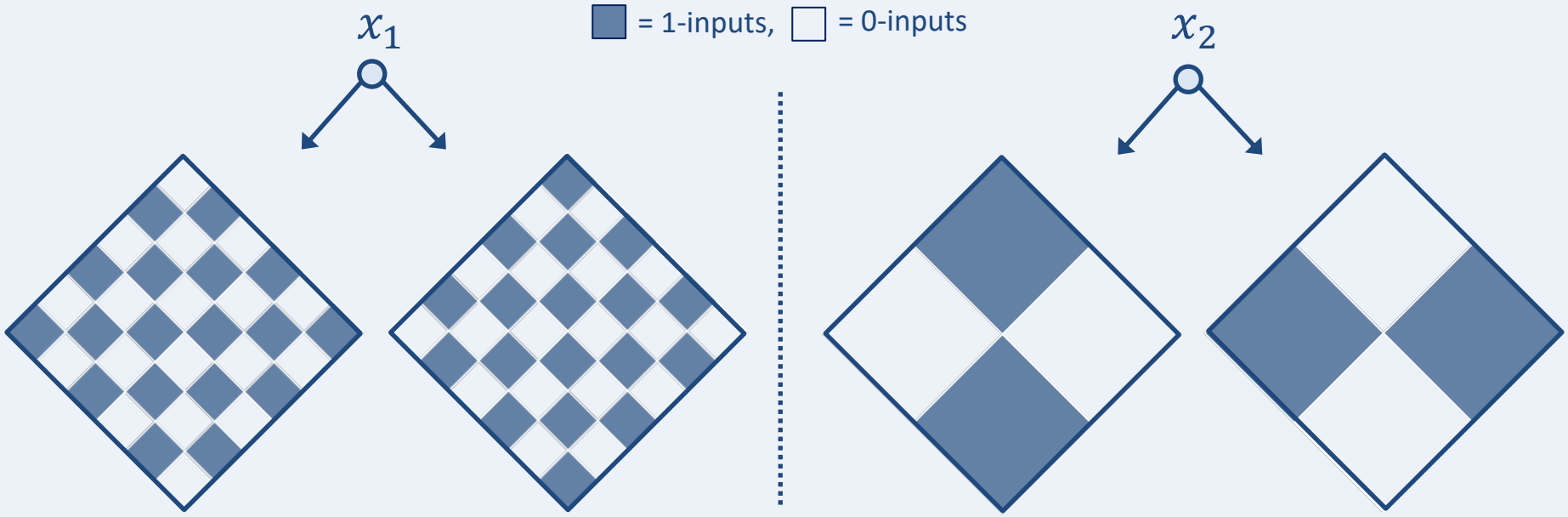
- Analogous identity for our new splitting criterion:

$$\sum_{\substack{S \ni i \\ |S| \le d}} (1-\delta)^{|S|} \cdot \mathbb{E}\left[f(x)\prod_{j \in S} x_j\right]^2 \eqsim \mathrm{NS}_\delta(f) - \mathbb{E}_b\left[\mathrm{NS}_\delta\left(f_{x_i=b}\right)\right]$$

Our algorithm greedily drives down the noise sensitivity of $f$

# Driving down variance vs. noise sensitivity

Consider two candidate splits:




- Expected **variance** of subfunctions same for both splits

  ∴ Both splits equally appealing to TopDown

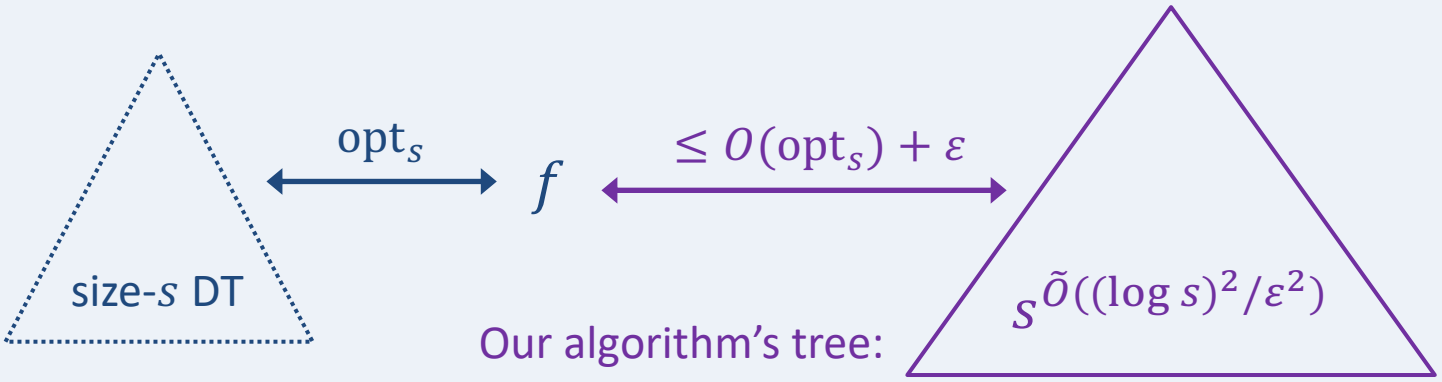
- Expected **noise sensitivity** of subfunctions much lower for $x_2$ split

  ∴ Our algorithm identifies $x_2$ as the better split

# End of Part II

A new splitting criterion and associated learning algorithm that achieves performance guarantees for all functions:

**Theorem:** For all $s \in \mathbb{N}$, $\varepsilon \in (0,1)$, and all target functions $f$,

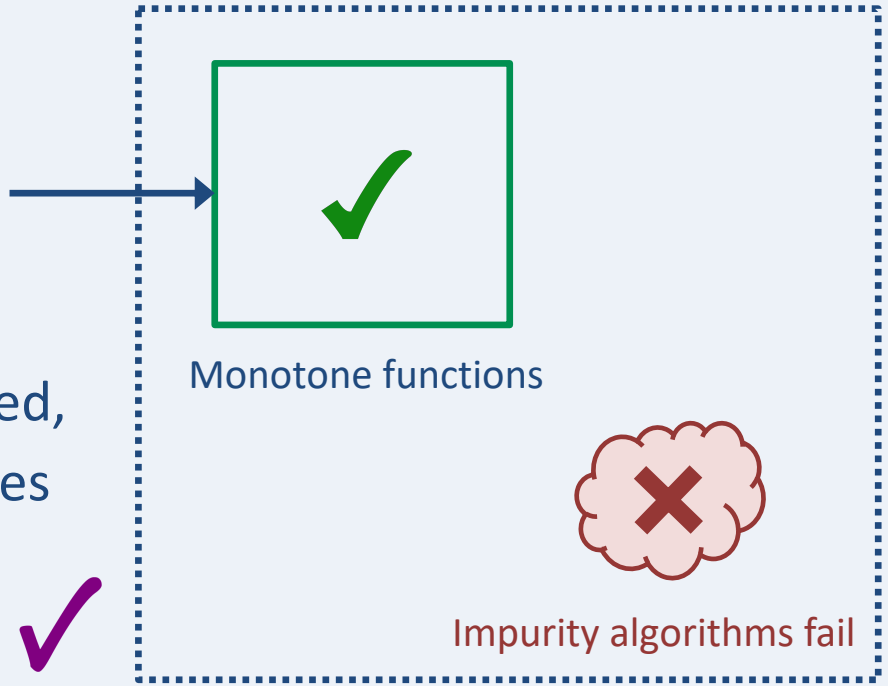


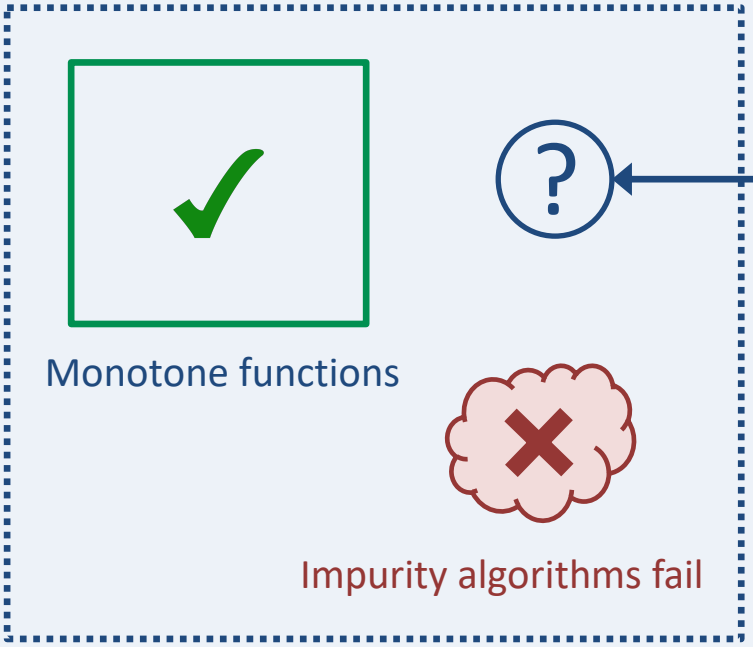

Our algorithm can be viewed as a "noise stabilization" procedure

# Recap of this talk

- How these top-down algorithms work: “impurity-based”
- **Bad news:** target functions for which impurity algorithms fail badly

- **Result I:** Performance guarantees for monotone functions
- **Result II:** A new, non-impurity-based, algorithm which achieves guarantees for all functions



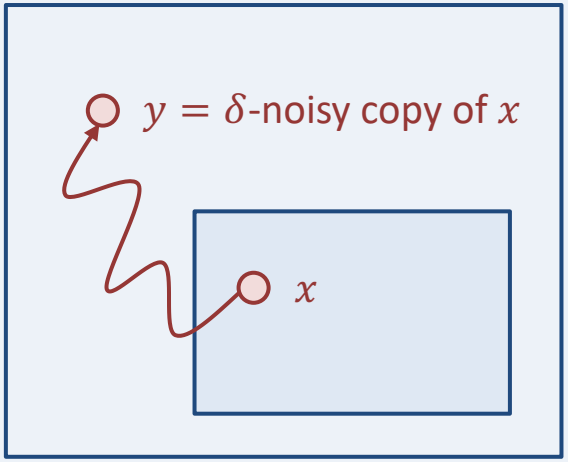



## Impurity algorithms:

- Guarantees for other broad and natural subclasses?
- Characterization?

## Beyond impurity algorithms:

- Understand power of new splitting criterion
- Noise sensitivity for other domains and distributions?

## Beyond a single tree:

- Random forests and boosted decision trees