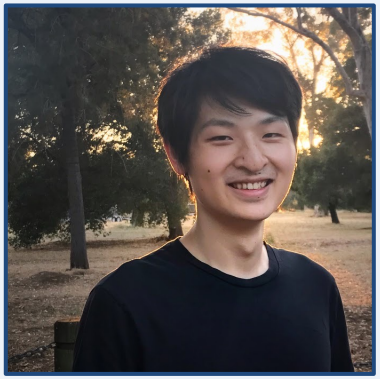
Properly learning decision trees in almost-polynomial time

Jane Lange

Joint work with:



Guy Blanc
Stanford

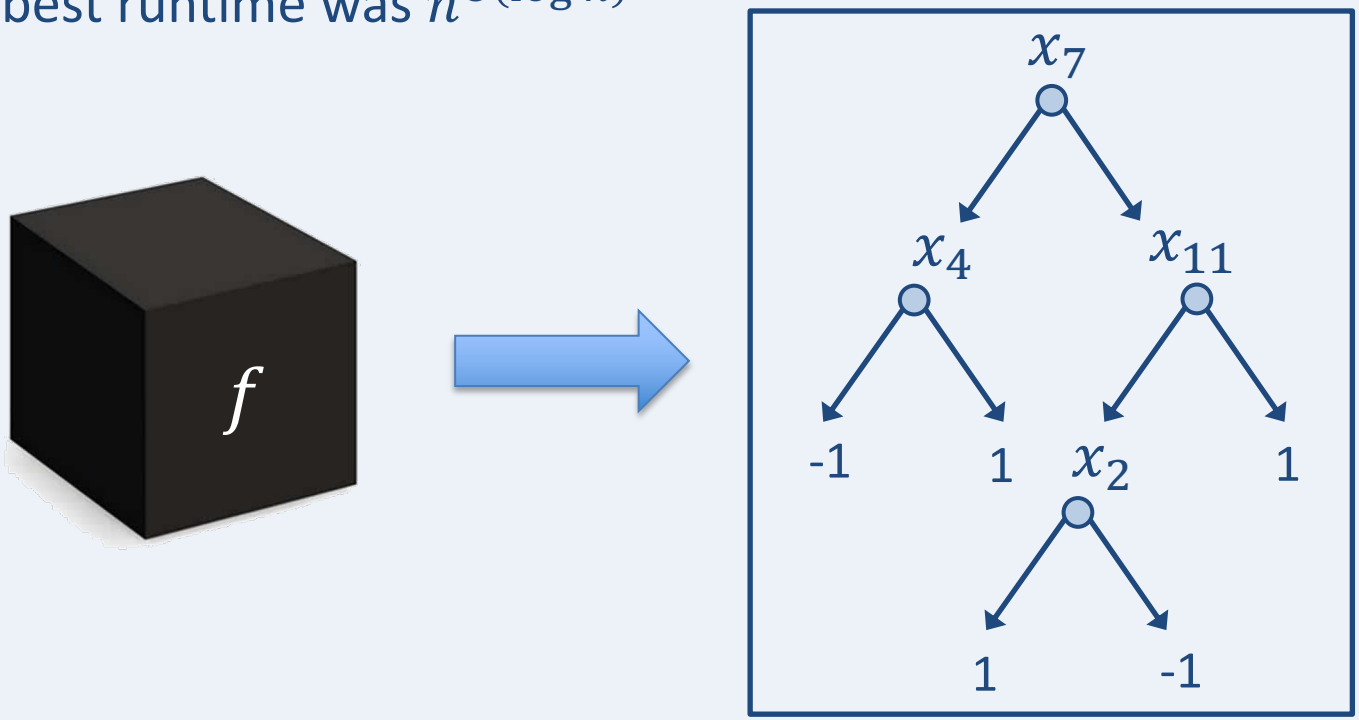
Mingda Qiao
Stanford

Li-Yang Tan
Stanford

Our main result

Our main result: A new algorithm for properly learning decision trees with membership queries in time $n^{O(\log \log n)}$

- Previous best runtime was $n^{O(\log n)}$



Query learning

Input: Query access to an unknown $f: \{-1,1\}^n \to \{-1,1\}$

Output: A hypothesis $h: \{-1,1\}^n \to \{-1,1\}$ that agrees with f on most inputs

w.r.t. uniform distribution

Characterizes this scenario:

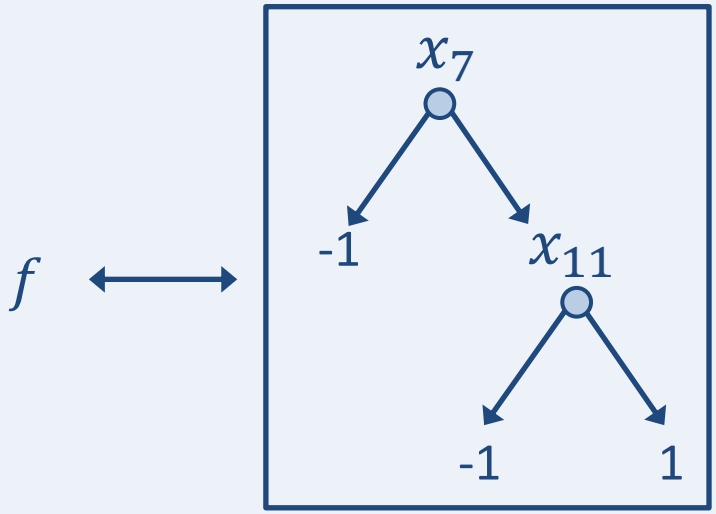
Each query access to f might be expensive.

1. Use a few queries to f to learn a good approximation, h, of f
2. Use h for future queries

Query learning decision trees

Input: Query access to an unknown $f: \{-1,1\}^n \to \{-1,1\}$

Assume f is close to a size-s decision tree



Output: A hypothesis $h: \{-1,1\}^n \to \{-1,1\}$ that agrees with f on most inputs

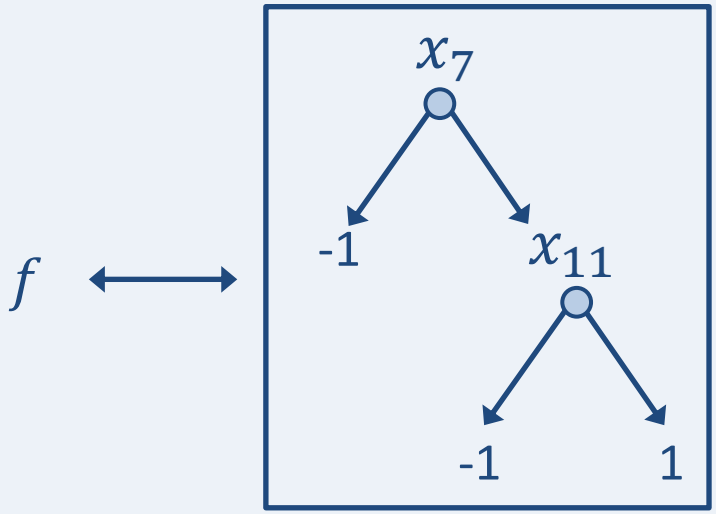
w.r.t. uniform distribution

Run in time $T(n, s)$

Properly query learning decision trees

Input: Query access to an unknown $f: \{-1,1\}^n \to \{-1,1\}$

Assume f is close to a size-s decision tree

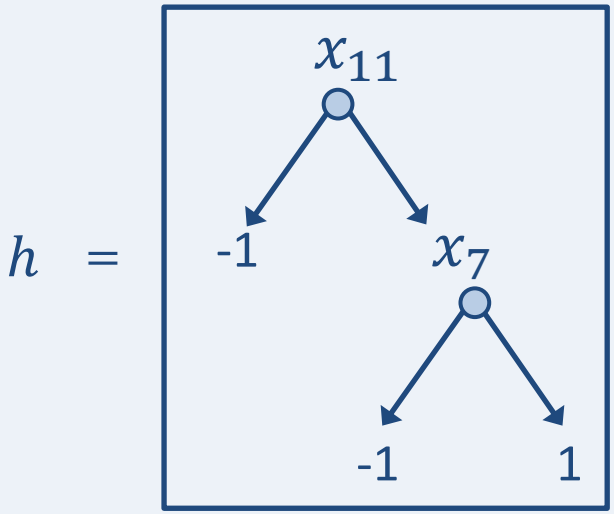


Output: A hypothesis $h: \{-1,1\}^n \to \{-1,1\}$ that agrees with f on most inputs

w.r.t. uniform distribution

Run in time $T(n, s)$

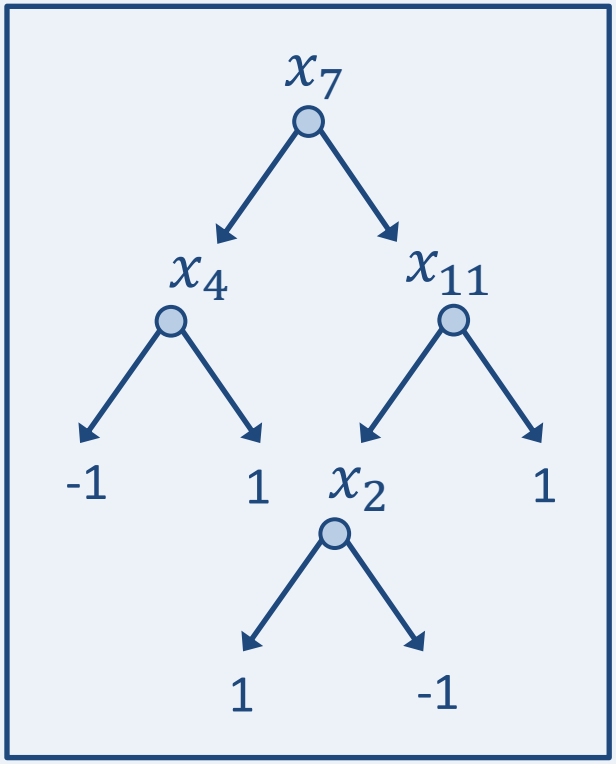
Output our hypothesis in a decision tree representation



Why a decision tree hypothesis?

We can turn any function f into the best decision tree representation!

1. DTs are the canonical example of explainable machine learning.
2. DTs are very fast to evaluate (relative to number of parameters).
3. DTs allow us to cheaply compute $f(x)$ when revealing features of x is expensive.



Our main result

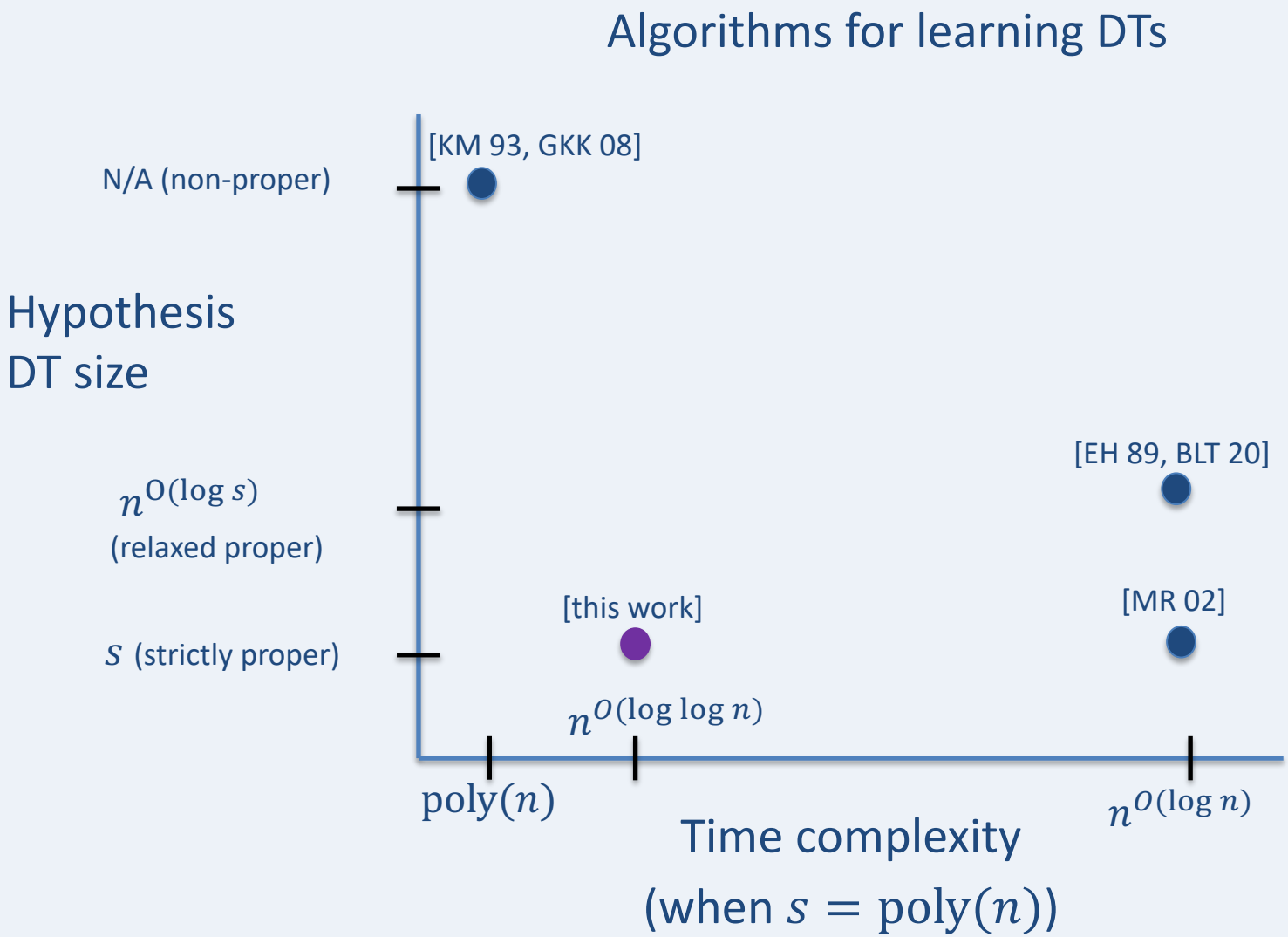
Theorem: Given query access to an unknown function f that is promised to be $\mathtt{opt}$ close to a size-s decision tree and $\varepsilon > 0$, our algorithm runs in time

$$\mathrm{poly}(n) \cdot s^{O_\varepsilon(\log \log s)}$$

and returns a decision tree hypothesis h that is $\mathtt{opt}+\varepsilon$-close to f.

- For $s = \mathrm{poly}(n)$, our runtime is $n^{O_\varepsilon(\log \log n)}$.
- Even for $\mathtt{opt} = 0$, previous best runtime was $n^{O_\varepsilon(\log n)}$ by variety of techniques [Ehrenfeucht-Haussler 89, Mehta-Raghavan 02, Blanc-Lange-Tan 20].
- Improper learning: Celebrated results give poly(n) time [Kushilevitz-Mansour 93, Gopalan-Kalai-Klivans 08].

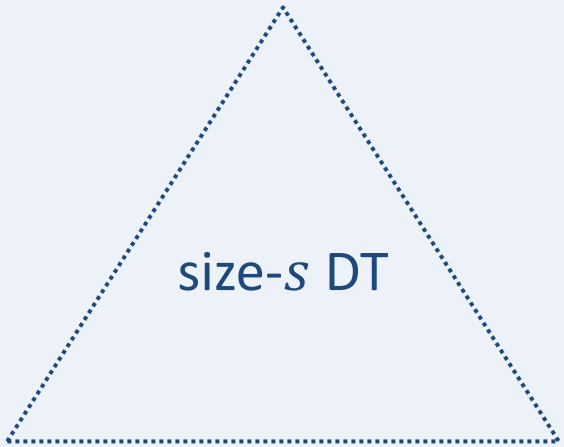
Our result in context



Part 1:

The realizable ($\mathtt{opt} = 0$) setting

Assume that:



$\equiv \quad f$

Realizable Rabbit icon indicates where we use this assumption

How [MR 02] works

High level idea: Create a subroutine that with the following input/output and call it recursively:

- Input: Some function $f: \{-1,1\}^n \to \{-1,1\}$ and a depth budget d.
- Output: A depth-d decision tree with minimal error w.r.t. f among all depth-d decision trees.

Fact: Let T be a size-s decision tree. For any $\varepsilon > 0$ and $d = \log(\frac{s}{\varepsilon})$, there is a depth-$d$ tree that is ε-close to T.

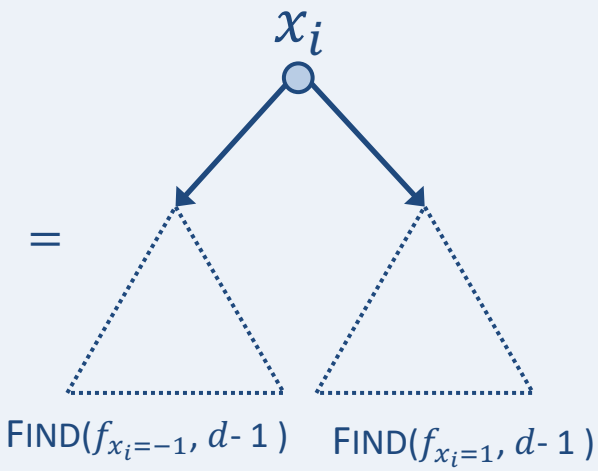
[MR 02] Pseudocode

FIND(f, d):

If $d = 0$, return the constant -1 or 1 function of minimal error.

For each i = 1, …, n, let

T_i =



Return the T_i with minimal error w.r.t. f.

[MR 02] Time complexity

- Each call to FIND with depth budget d makes $2n$ recursive calls to FIND with depth budget $d - 1$
- Number calls = $(2n)^d = n^{O(\log s)}$

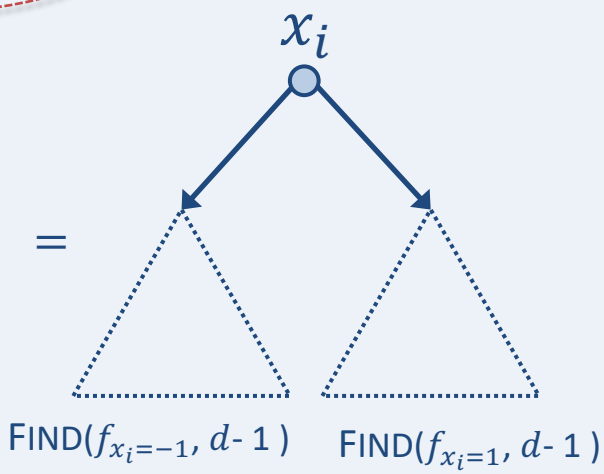
Our main idea

FIND(f, d):

If $d = 0$, return the constant -1 or 1 function of minimal error.

For each i = 1, …, n, let

T_i =



Return the T_i with minimal error w.r.t. f.

Influence as a heuristic

- Instead of brute force searching over all n possible roots, only search over select few "important" possible roots.

Definition: The influence of x_i with respect to a function f is

$$\mathrm{inf}_i(\mathrm{f}) := \Pr_{\boldsymbol{x} \sim \{-1,1\}^n}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\sim i})\right]$$

Influences can be efficiently measured from queries to f.

This means $\boldsymbol{x}$ with the ith bit rerandomized

Known fact: If f = depth-d decision tree, sum of influences $\leq \mathrm{d}$.

Corollary: If f = depth-d decision tree, $\leq \frac{d}{\tau}$ variables with influence $\geq \tau$.

Our pseudocode

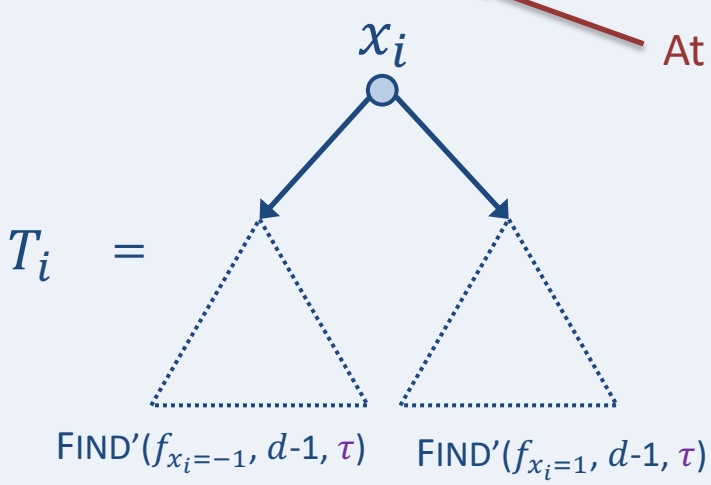
Main lemma: τ can be set to $\frac{\varepsilon}{d}$

FIND'(f, d, τ):

If $d = 0$, return the constant 0 or 1 function of minimal error.

For each i = 1, …, n satisfying $\mathrm{inf}_i(\mathrm{f}) > \tau$,

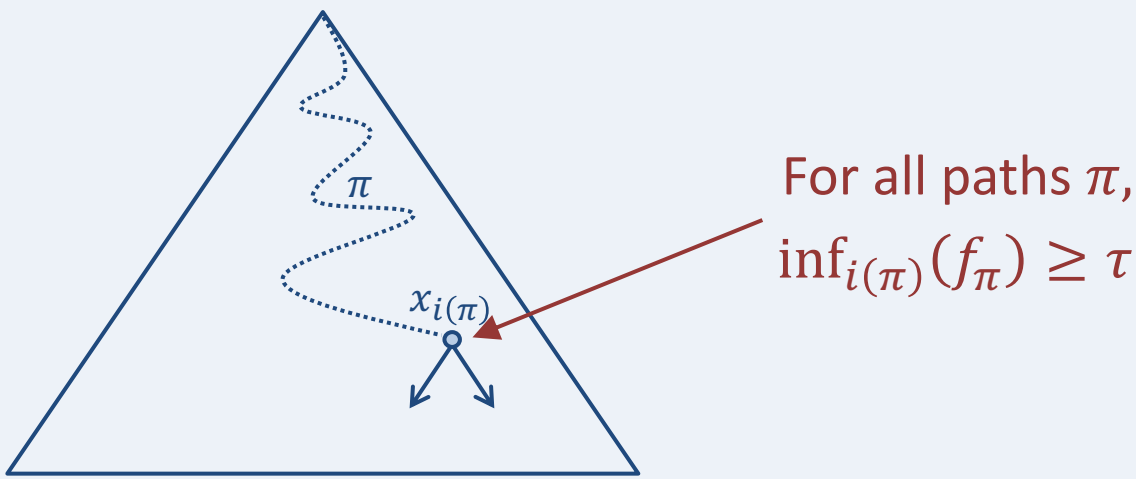
At most $\frac{d^2}{\varepsilon}$ such variables



Return the T_i with minimal error.

Which trees does our algorithm search over?

Definition: For any $f: \{-1,1\}^n \to \{-1,1\}$, decision tree T, and $\tau > 0$, we say that T is everywhere τ-influential w.r.t. f if,



Fact: FIND'(f, d, τ) returns a depth-d DT with minimal error among all depth-d DTs that are everywhere τ-influential w.r.t. f

Our time complexity

- Each call to FIND' with depth budget d makes d/τ recursive calls to FIND' with depth budget $d-1$ 
- Number of calls $= (d/\tau)^d$

Still to show: τ can be set to $\frac{\varepsilon}{d}$ and we recover a good decision tree.

- Assuming this, $(d/\tau)^d \approx (d^2)^d \approx (\log s)^{O(\log s)} = s^{O(\log\log s)}$
- Time complexity is $O(n) \cdot s^{O(\log\log s)}$

Main structural theorem

Theorem: For any $f: \{-1,1\}^n \to \{-1,1\}$, $\varepsilon > 0$, and depth-d DT T, there is a DT T' with depth at most d that is $\frac{\varepsilon}{d}$-influential w.r.t. f satisfying

$$\mathrm{dist}(T', f) \leq \mathrm{dist}(T, f) + \varepsilon$$

Every decision tree can be "pruned" so that *every* variable has high influence.

This is a generalization of a famous inequality by O'Donnell, Saks, Schramm, and Servedio 05

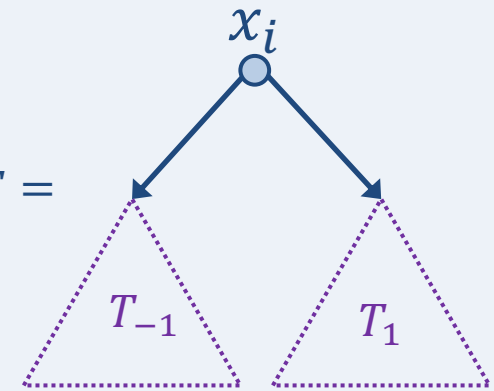
Theorem [OSSS 05]: For any $f: \{0,1\}^n \to \{0,1\}$ computed by a depth-d decision tree, there is an $i \in [n]$ such that:

$$\mathrm{inf}_i(f) \geq \frac{\mathrm{Var}(f)}{d}$$

Every decision tree has an influential variable.

Pruning defined recursively

$T =$

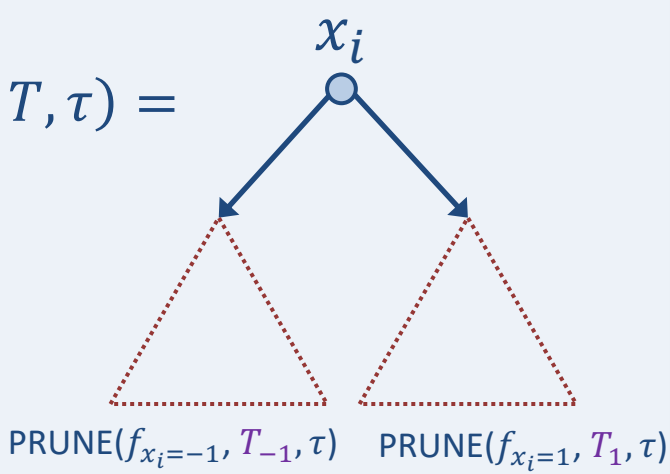


Base case: If T has depth 0:

$$\text{PRUNE}(f, T, \tau) = T$$

If $\inf_i(f) \geq \tau$: Do not prune root

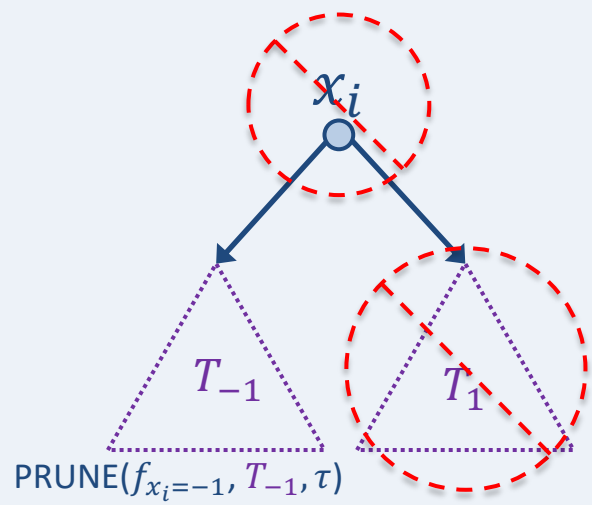
$\text{PRUNE}(f, T, \tau) =$



If $\inf_i(f) < \tau$: Prune root

$\text{PRUNE}(f, T, \tau) =$ whichever is closer to f:

$\text{PRUNE}(f, T_{-1}, \tau)$ or $\text{PRUNE}(f, T_1\ \tau)$



Pruning proof

Lemma: For any $f: \{-1,1\}^n \to \{-1,1\}$, $\tau > 0$, and depth-d DT T,

$$\mathrm{dist}(f, \mathrm{PRUNE}(f, T, \tau)) \leq \mathrm{dist}(f, T) + d \cdot \tau$$

Proof: By induction on d. Base case of $d = 0$ is easy.

For $d \geq 1$, we only do pruning case here (it's harder):

$$\mathrm{dist}(f, \mathrm{PRUNE}(f, T, \tau)) \leq \mathbb{E}_{\mathrm{b} \sim \{-1,1\}}\left[\mathrm{dist}\left(f, \mathrm{PRUNE}(f, T_b, \tau)\right)\right] \qquad \min \leq \mathrm{mean}$$

$$\leq \mathbb{E}_{\mathrm{b} \sim \{-1,1\}}\left[\mathrm{dist}(f, T_b) + (d-1) \cdot \tau\right] \qquad \text{Inductive hyp}$$

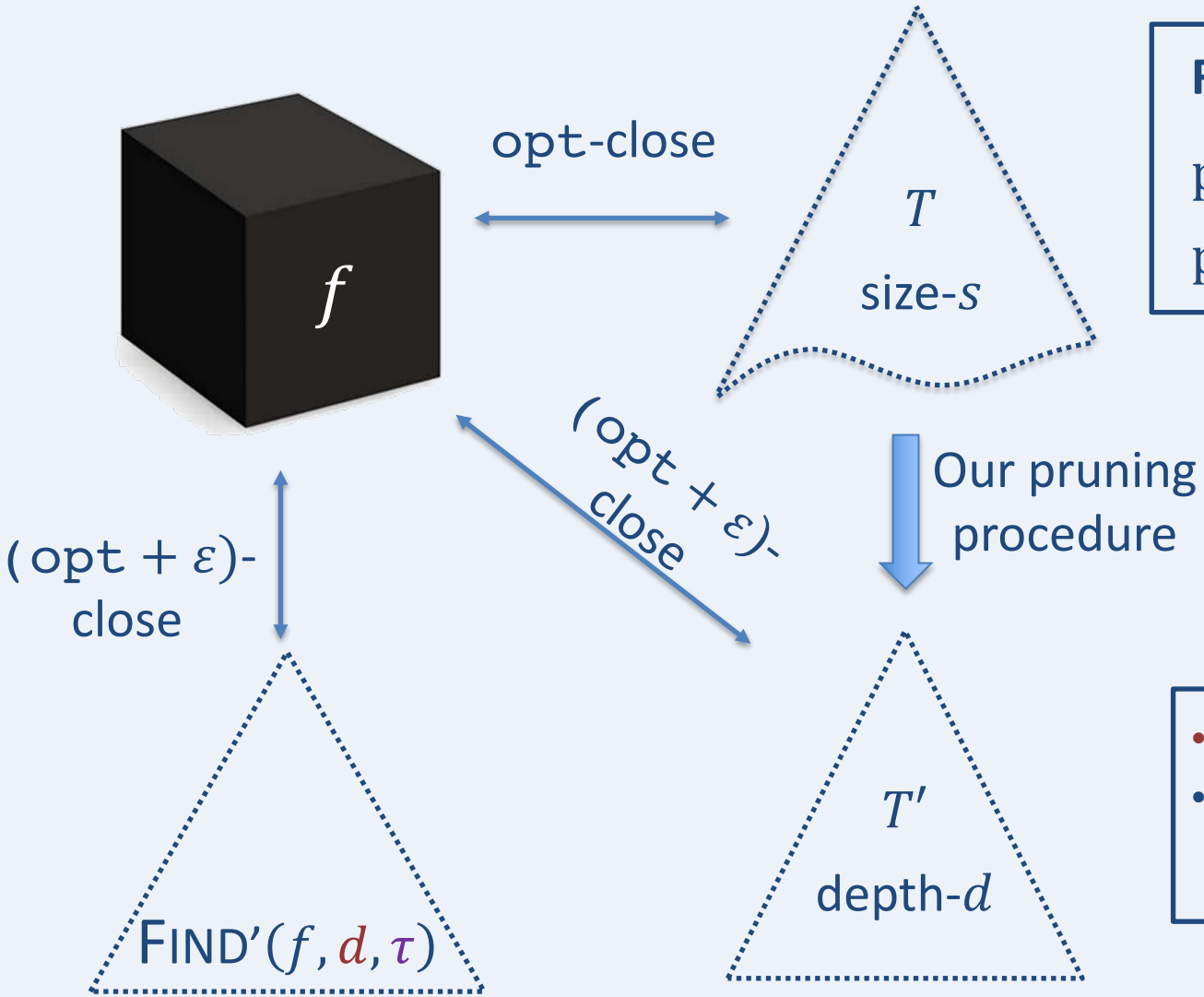
$$\leq \mathbb{E}_{\mathrm{b} \sim \{-1,1\}}\left[\mathrm{dist}\left(f, f_{x_i=b}\right) + \mathrm{dist}\left(f_{x_i=b}, T_b\right) + (d-1) \cdot \tau\right] \qquad \text{Triangle ineq}$$

$$= \mathbb{E}_{\mathrm{b} \sim \{-1,1\}}\left[\mathrm{inf}_i(f) + \mathrm{dist}\left(f_{x_i=b}, T_b\right) + (d-1) \cdot \tau\right] \qquad \text{Def of inf}$$

$$= \mathrm{inf}_i(f) + \mathrm{dist}(f, T) + (d-1) \cdot \tau \qquad \text{Law of total exp.}$$

$$\leq \mathrm{dist}(f, T) + d \cdot \tau \qquad \mathrm{inf}_i(f) \leq \tau$$

Putting everything together



Runtime:

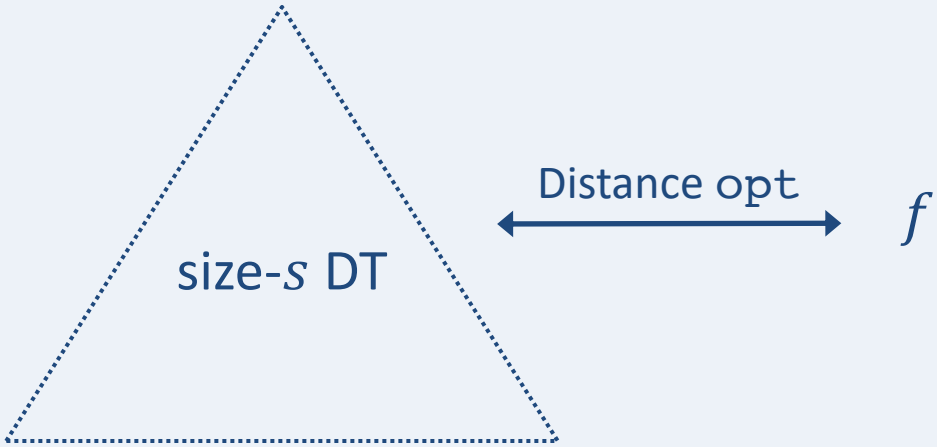
$$\mathrm{poly}(n) \cdot \left(\frac{d}{\tau}\right)^d = \mathrm{poly}(n) \cdot s^{O_\varepsilon(\log\log s)}$$

- $d := \log(s/\varepsilon)$
- T' is $\tau := \frac{\varepsilon}{d}$-influential w.r.t. f

Part 2:

The agnostic ($\mathtt{opt} > 0$) setting

Now:



Challenge: No longer guaranteed that f has $\leq \frac{d}{\tau}$ variables with influence $\geq \tau$.

Special case: monotone f

Definition: A function $f: \{-1,1\}^n \to \{-1,1\}$ is monotone if

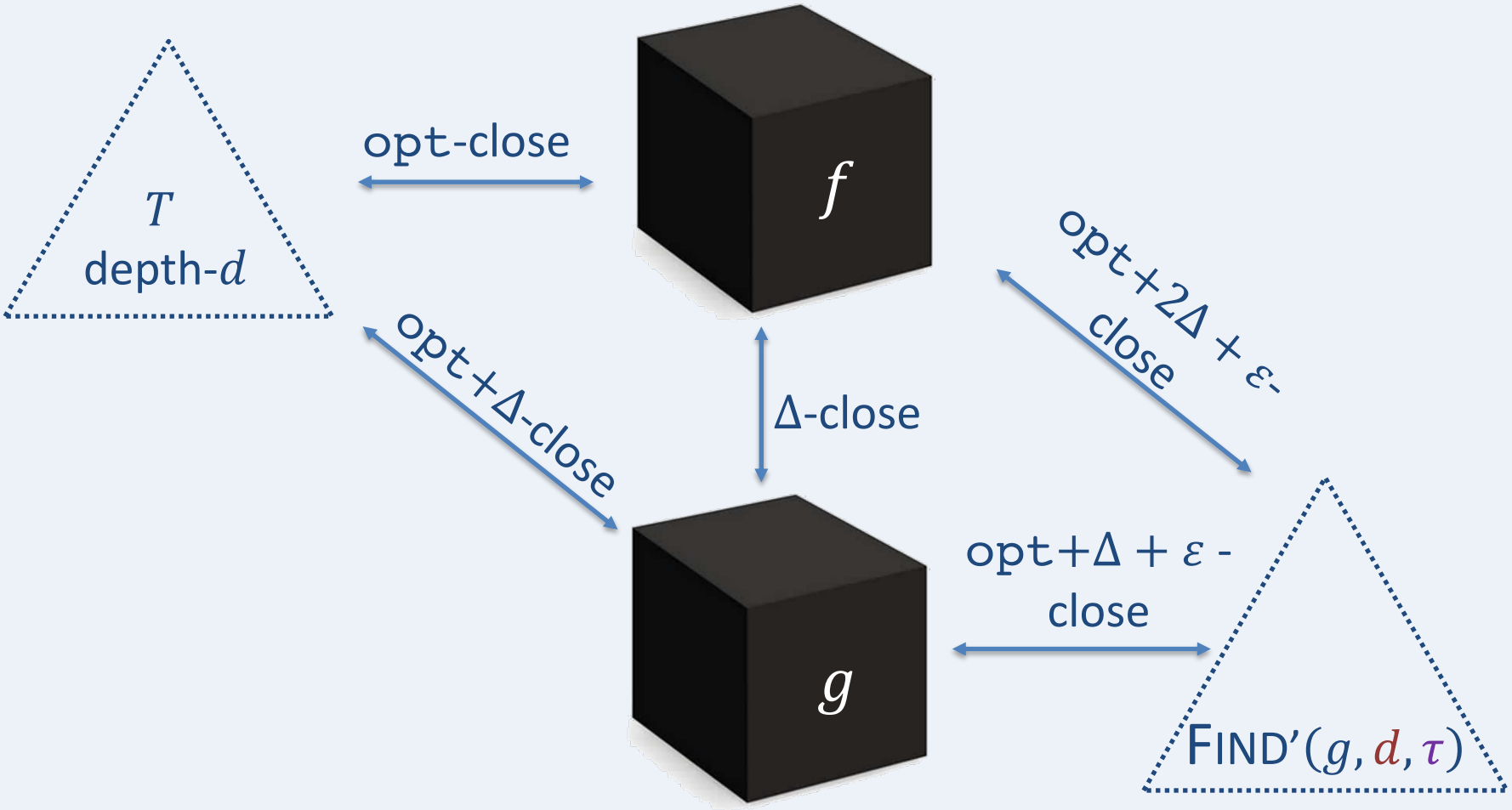
$$\forall i. x_i \geq y_i \to f(x) \geq f(y)$$

Fact: For any monotone f, $\sum_{i \in [n]} \inf_i(f) \leq 1$

Then there are $\leq \frac{1}{\tau}$ variables with influence $\geq \tau$.

Fact: If f is monotone, influences can be estimated from random examples labeled by f.

Key idea: Use a "smooth" function close to f



g has few highly influential variables

Requirements of our smooth function

1. We want g to be close to f
2. We want to be able to efficiently simulate queries to g given query access to f
3. We want g to have few variables of high influence

Definition: For function f and $\delta \in (0,1)$, we define "smoothed f" as

$$f_\delta(x) := \mathbb{E}_{\boldsymbol{y} \underset{\delta}{\sim} x}\left[f(\boldsymbol{y})\right]$$

$\boldsymbol{y}$ generated by starting with x and independently rerandomizing each bit with probability δ

Properties of smoothed functions

Trees are smooth. For any depth-d decision tree $T: \{-1,1\}^n \to \{-1,1\}$ and $\delta \in (0,1)$,

$$\operatorname{dist}(T_\delta, T) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \{-1,1\}^n} [|T_\delta(\boldsymbol{x}) - T(\boldsymbol{x})|] \leq \delta \cdot d$$

Equivalent to noise sensitivity of T

Via sum of influences

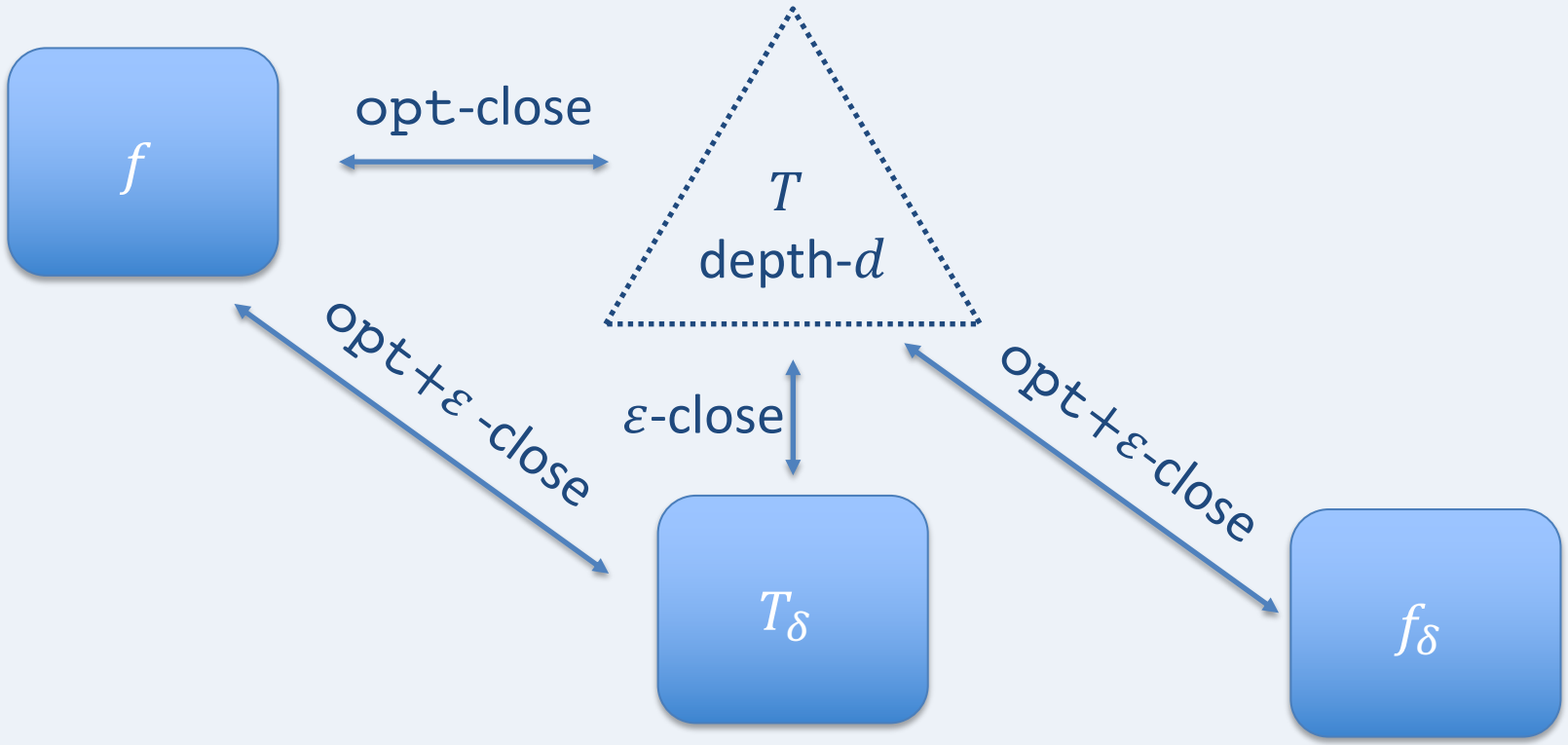
Symmetry. For any functions $f, g: \{-1,1\}^n \to \{-1,1\}$, and $\delta \in (0,1)$,

$$\operatorname{dist}(f_\delta, g) = \operatorname{dist}(f, g_\delta)$$

Facts about smoothed functions

Corollary: If f is opt close to a depth-d DT T, then for $\delta = \frac{\varepsilon}{d}$

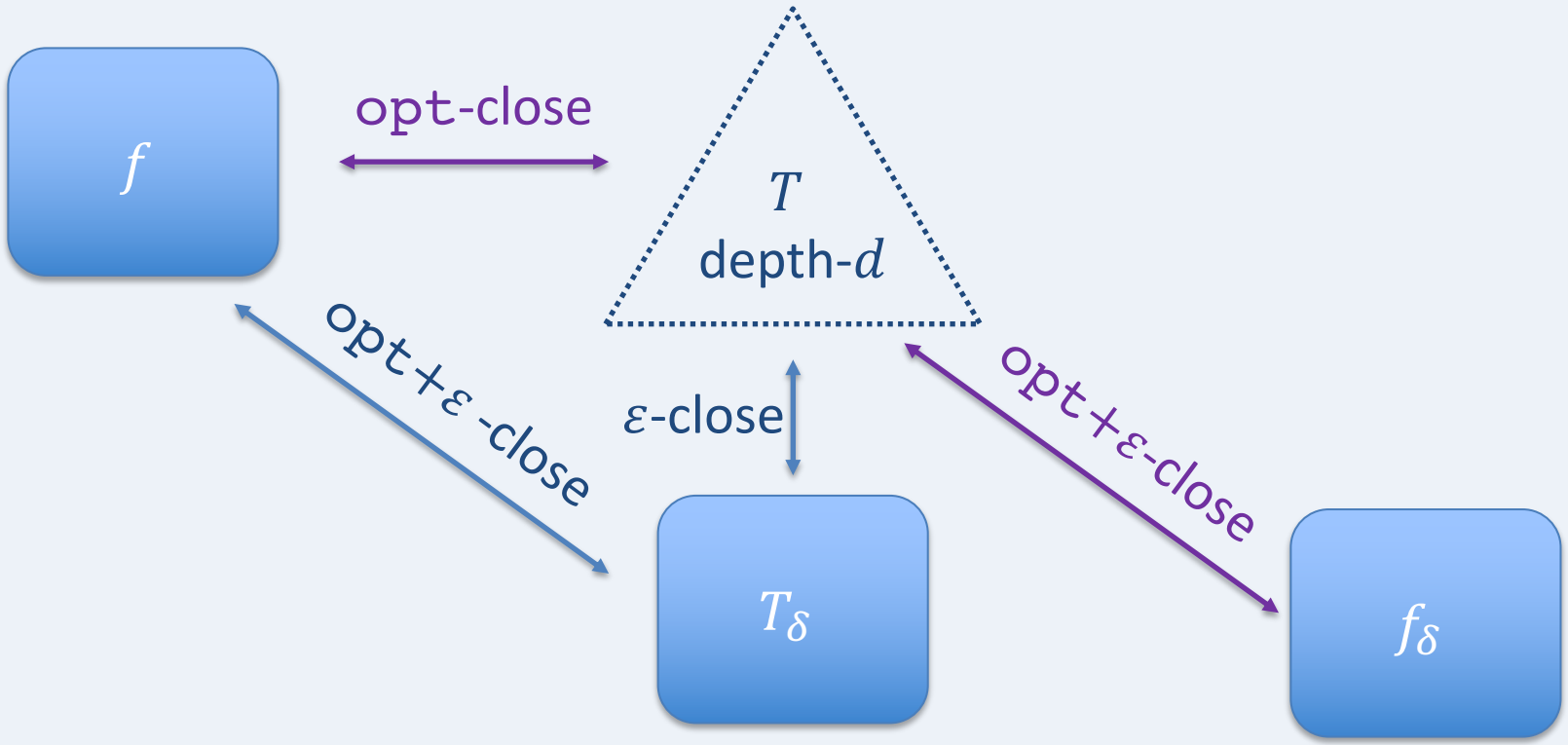
$$\mathrm{dist}(f, f_\delta) \leq 2 \cdot \mathtt{opt} + \varepsilon$$



Facts about smoothed functions

Corollary: If f is opt close to a depth-d DT T, then for $\delta = \frac{\varepsilon}{d}$

$$\mathrm{dist}(f, f_\delta) \leq 2 \cdot \mathtt{opt} + \varepsilon$$



Requirements of our smooth function

1. We want f_δ to be close to f ☑
2. We want to be able to efficiently simulate queries to f_δ given query access to f
3. We want f_δ to have few variables of high influence

Corollary: If f is $\texttt{opt}$ close to a depth-d DT T, then for $\delta = \frac{\varepsilon}{d}$

$$\mathrm{dist}(f, f_\delta) \leq 2 \cdot \texttt{opt} + \varepsilon$$

Requirements of our smooth function

1. We want f_δ to be close to f ☑
2. We want to be able to efficiently simulate queries to f_δ given query access to f ☑
3. We want f_δ to have few variables of high influence

Definition: For any function f and $\delta \in (0,1)$, we define "smooth f" as

$$f_\delta(x) := \mathbb{E}_{\boldsymbol{y} \underset{\delta}{\sim} x}[f(\boldsymbol{y})]$$

Influence for real-valued functions

Two possible definitions of influence for $f: \{-1,1\}^n \to \mathbb{R}$

Definition: The L_2 influence of x_i with respect to a function f is

$$\mathrm{inf}_i^{(2)}(f) := \mathop{\mathbb{E}}_{\boldsymbol{x}}\left[\left(f(\boldsymbol{x}) - f\left(\boldsymbol{x}^{\sim i}\right)\right)^2\right]$$

Pro: Easy to analyze (Fourier friendly)

Con: Pruning proof doesn't work because $d(x,y) = (x-y)^2$ not a metric

Definition: The L_1 influence of x_i with respect to a function f is

$$\mathrm{inf}_i^{(1)}(f) := \mathop{\mathbb{E}}_{\boldsymbol{x}}\left[\left|f(\boldsymbol{x}) - f\left(\boldsymbol{x}^{\sim i}\right)\right|\right]$$

Pro: Pruning works because $d(x,y) = |x - y|$ is a metric

Con: Harder to analyze directly (but we can relate it to L_2 influence)

Influence of f_δ

Desired property : For any $f: \{-1,1\}^n \to \{-1,1\}$ and $\tau, \delta \in (0,1)$, there are $\leq \frac{1}{\delta \cdot \tau^2}$ variables with L_1 influence $\geq \tau$ w.r.t. f_δ.

Fact 1: For any $f: \{-1,1\}^n \to \{-1,1\}$ and $\delta \in (0,1)$

$$\sum_{i=1}^{n} \inf_i^{(2)}(f_\delta) \leq \frac{1}{\delta}$$

Proof is easy using Fourier analysis.

Fact 2: For any $f: \{-1,1\}^n \to \mathbb{R}$ and variable x_i

$$\inf_i^{(1)}(f) \leq \sqrt{\inf_i^{(2)}(f)}$$

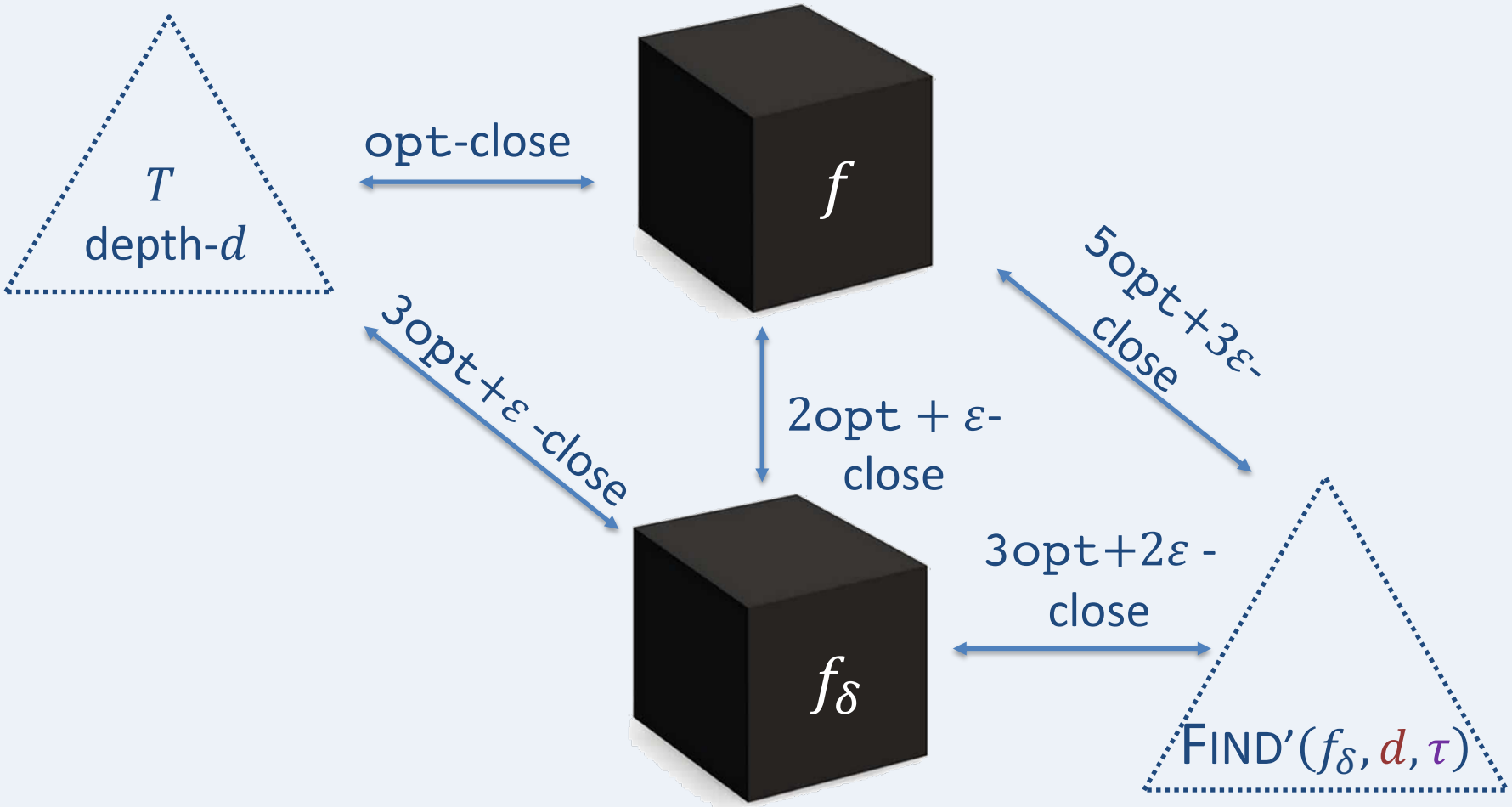
Proof: Jensen's inequality

Requirements of our smooth function

1. We want f_δ to be close to f ☑
2. We want to be able to efficiently simulate approximate queries to f_δ given query access to f ☑
3. We want f_δ to have few variables of high influence ☑

Property : For any $f: \{-1,1\}^n \to \{-1,1\}$ and $\tau, \delta \in (0,1)$, there are $\leq \frac{1}{\delta \cdot \tau^2}$ variables with L_1 influence $\geq \tau$ w.r.t. f_δ.

Putting everything together



f_δ has few highly influential variables

We've proven

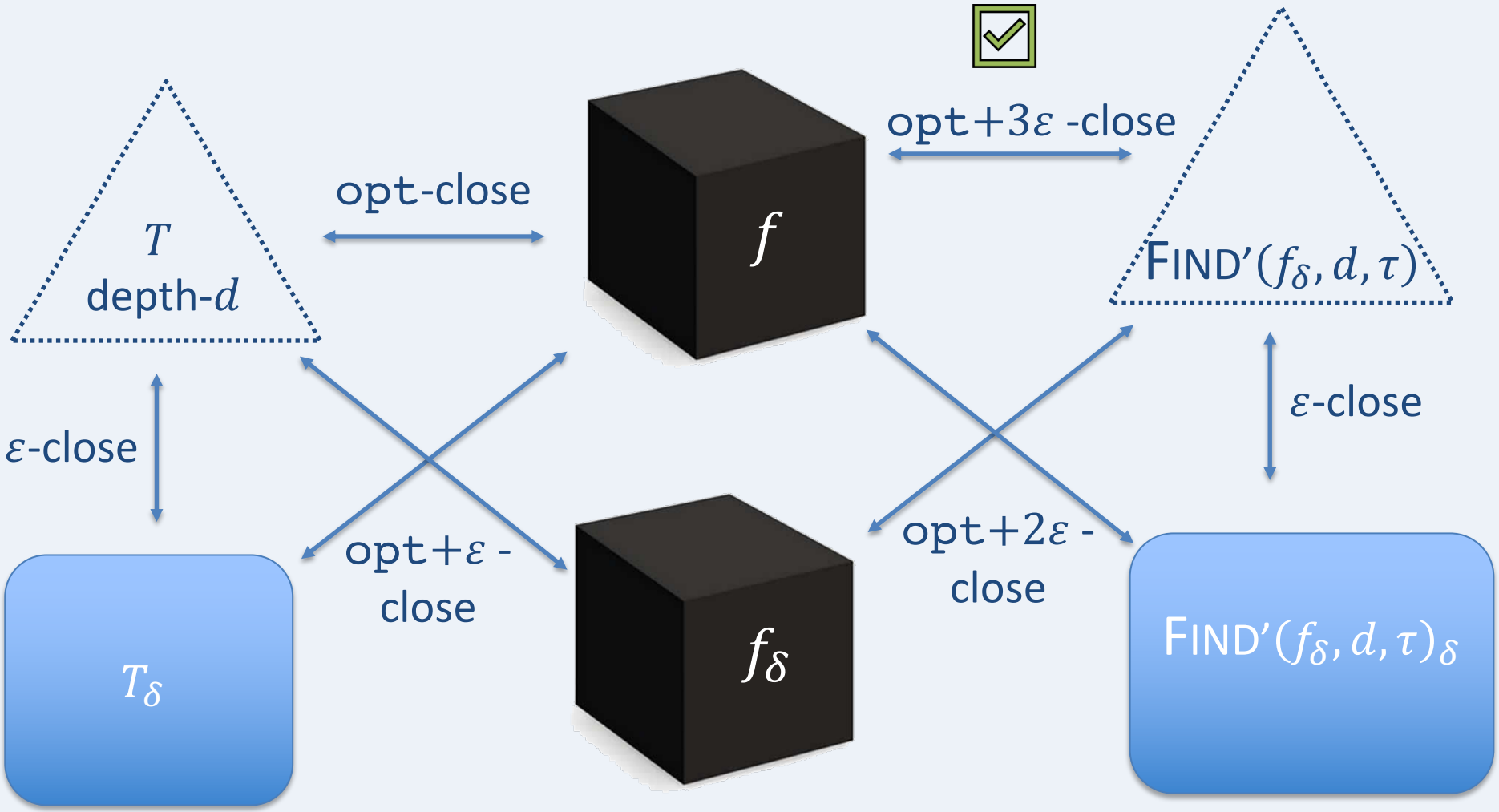
Theorem: Given query access to an unknown function f that is promised to be $\mathtt{opt}$ close to a size-s decision tree and $\varepsilon > 0$, our algorithm runs in time

$$\mathrm{poly}(n) \cdot s^{O_\varepsilon(\log \log s)}$$

and returns a decision tree hypothesis h that is $5 \cdot \mathtt{opt} + \varepsilon$-close to f.

Can we turn $5 \cdot \mathtt{opt}$ into $1 \cdot \mathtt{opt}$?

More careful analysis: $5 \cdot \texttt{opt} \to 1 \cdot \texttt{opt}$



Bonus slides: exact size

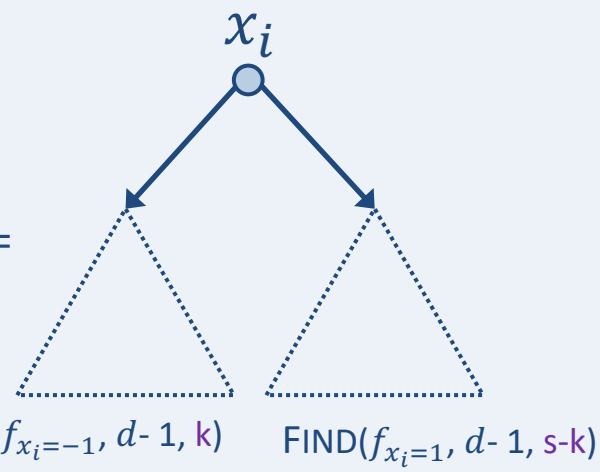
FIND(f, d, s):

Return cached tree if it is nonempty.

If $d = 0$ or $s = 1$, return the constant -1 or 1 function of minimal error.

For each $i : \mathrm{inf}_i(\mathrm{f}) > \tau$, and k = 1, …, $s - 1$, let

$T_{i,k} =$



Return and cache the $T_{i,k}$ with minimal error w.r.t. f.

Bonus slides: exact size

- Bound the total number of calls to FIND'.
- We reach the recursive step at most once per pair (π, k)
- Number of pairs = $(d/\tau)^d \cdot 2^d = s^{O(\log \log s)}$
- Number of calls per pair = $2s \ * |\{i \mid \mathrm{inf}_i(\mathrm{f}) > \tau\}|$
- Runtime is still $s^{O(\log \log s)}$

Our result in context