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: 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 \(f: \{-1,1\}^n \rightarrow \{-1,1\}\)

Output: A hypothesis \(h: \{-1,1\}^n \rightarrow \{-1,1\}\) 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 $f: \{-1,1\}^n \rightarrow \{-1,1\}$

$x \xrightarrow{} f \xrightarrow{} f(x)$

Assume $f$ is close to a size-$s$ decision tree

Output: A hypothesis $h: \{-1,1\}^n \rightarrow \{-1,1\}$ 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 \( f: \{-1,1\}^n \rightarrow \{-1,1\} \)

\( x \xrightarrow{} f \xrightarrow{} f(x) \)

Assume \( f \) is close to a size-\( s \) decision tree

Output: A hypothesis \( h: \{-1,1\}^n \rightarrow \{-1,1\} \) 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

\[ h = \]

\[ f = \]
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 $\text{opt}$ close to a size-$s$ decision tree and $\varepsilon > 0$, our algorithm runs in time

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

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

- For $s = \text{poly}(n)$, our runtime is $n^{O_\varepsilon(\log \log n)}$.

- Even for $\text{opt} = 0$, 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 $\text{poly}(n)$ time [Kushilevitz-Mansour 93, Gopalan-Kalai-Klivans 08].
Our result in context

Algorithms for learning DTs

Hypothesis

DT size

Time complexity

(when $s = \text{poly}(n)$)

\[ n^O(\log s) \] (relaxed proper)

\[ n \] (strictly proper)

poly($n$)

[KM 93, GKK 08]

N/A (non-proper)

[EH 89, BLT 20]

[this work]

[MR 02]

\[ n^{O(\log \log n)} \]
Part 1:
The realizable (opt = 0) setting

Assume that: size-\(s\) DT \(\equiv f\)

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 \( f : \{-1,1\}^n \rightarrow \{-1,1\} \) 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 \( \varepsilon \)-close to \( T \).
**Pseudocode**

**FIND**(\(f, d\)):

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

- For each \(i = 1, ..., n\), let

\[
T_i = \text{FIND}(f_{x_i=-1}, d-1) \quad \text{and} \quad \text{FIND}(f_{x_i=1}, d-1)
\]

- 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

\[
T_i = \begin{cases} 
\text{FIND(} f_{x_i=-1}, d-1 \text{)} & \text{if } x_i = -1 \\
\text{FIND(} f_{x_i=1}, d-1 \text{)} & \text{if } x_i = 1 
\end{cases}
\]

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

$$\inf_i (f) := \Pr \left[ f(x) \neq f(x^{\sim i}) \right]$$

$x \sim \{−1,1\}^n$

Influences can be efficiently measured from queries to $f$.

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

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

\[ \text{Main lemma: \( \tau \) can be set to } \frac{\varepsilon}{d} \]

FIND'(\( f, d, \tau \)):

If \( d = 0 \), return the constant 0 or 1 function of minimal error.

For each \( i = 1, \ldots, n \) satisfying \( \inf_i f > \tau \),

\[ T_i = \begin{cases} 
\text{FIND}'(f_{x_i=-1}, d-1, \tau) & \text{if } f_{x_i=-1} \leq \tau \\
\text{FIND}'(f_{x_i=1}, d-1, \tau) & \text{if } f_{x_i=1} > \tau 
\end{cases} \]

Return the \( T_i \) with minimal error.
Which trees does our algorithm search over?

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

$$\inf_{i(\pi)}(f_\pi) \geq \tau$$

**Fact:** $\text{FIND}'(f, d, \tau)$ returns a depth-$d$ DT with minimal error among all depth-$d$ DTs that are everywhere $\tau$-influential w.r.t. $f$
Our time complexity

- Each call to \texttt{FIND'} with depth budget \( d \) makes \( d/\tau \) recursive calls to \texttt{FIND'} with depth budget \( d - 1 \).
- Number of calls = \( (d/\tau)^d \)

\textbf{Still to show:} \( \tau \) can be set to \( \frac{\epsilon}{d} \) 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 \( O(n) \cdot s^{O(\log \log s)} \)
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
\[
\text{dist}(T', f) \leq \text{dist}(T, f) + \varepsilon
\]

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:
\[
\inf_i(f) \geq \frac{\text{Var}(f)}{d}
\]

Every decision tree has an influential variable.
Pruning defined recursively

Base case: If $T$ has depth 0:

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

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

$$\text{PRUNE}(f, T, \tau) = \begin{cases} x_i & \text{if } \inf_i(f) \geq \tau \text{ in } T_i \text{ and } \inf_i(f) \geq \tau \text{ in } T_{-1} \text{ or } T_1 \text{, closer to } f \text{ in } T_{-1} \text{ or } T_1 \end{cases}$$

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

$$\text{PRUNE}(f, T, \tau) = \text{whichever is closer to } f: \text{PRUNE}(f, T_{-1}, \tau) \text{ or PRUNE}(f, T_1, \tau)$$
Pruning proof

Lemma: For any $f: \{-1,1\}^n \to \{-1,1\}$, $\tau > 0$, and depth-$d$ DT $T$,  
$$ \text{dist}(f, \text{PRUNE}(f, T, \tau)) \leq \text{dist}(f, T) + d \cdot \tau $$

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):

$$ \text{dist}(f, \text{PRUNE}(f, T, \tau)) \leq \mathbb{E}_{b \sim \{-1,1\}}[\text{dist}(f, \text{PRUNE}(f, T_b, \tau))] $$

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

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

$$ = \mathbb{E}_{b \sim \{-1,1\}}[\inf_i(f) + \text{dist}(f_{x_i=b}, T_b) + (d - 1) \cdot \tau] $$

$$ = \inf_i(f) + \text{dist}(f, T) + (d - 1) \cdot \tau $$

$$ \leq \text{dist}(f, T) + d \cdot \tau $$
Putting everything together

\[ \begin{align*}
T & \quad \text{size-}s \\
T' & \quad \text{depth-}d \\
\text{FIND'}(f, d, \tau) & \quad (\text{opt + \(\varepsilon\))-close}
\end{align*} \]

**Runtime:**
\[
\text{poly}(n) \cdot \left( \frac{d}{\varepsilon} \right)^d = \text{poly}(n) \cdot s^{O_{\varepsilon} \log \log s}
\]

- \(d := \log(s/\varepsilon)\)
- \(T'\) is \(\varepsilon\)-influential w.r.t. \(f\)
Part 2: The agnostic ($\text{opt} > 0$) setting

Now:

size-$s$ DT

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

Special case: monotone $f$

**Definition:** A function $f : \{-1,1\}^n \rightarrow \{-1,1\}$ is monotone if
$$\forall i. \ x_i \geq y_i \rightarrow f(x) \geq f(y)$$

**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

$T_{\text{depth}-d}$

$\text{opt-close}$

$\text{opt} + \Delta$-close

$\Delta$-close

$\text{opt} + 2\Delta + \varepsilon$-close

$\text{opt} + \Delta + \varepsilon$-close

$\text{FIND}'(g, d, \tau)$
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

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

$y$ generated by starting with $x$ and independently rerandomizing each bit with probability $\delta$
**Properties of smoothed functions**

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

$$dist(T_\delta, T) := \mathbb{E} \left[ |T_\delta(x) - T(x)| \right] \leq \delta \cdot d$$

- Equivalent to noise sensitivity of $T$
- Via sum of influences

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

$$dist(f_\delta, g) = dist(f, g_\delta)$$
Facts about smoothed functions

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

\[
\text{dist}(f, f_\delta) \leq 2 \cdot \text{opt+}\varepsilon
\]
Facts about smoothed functions

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

\[
\text{dist}(f, f_\delta) \leq 2 \cdot \text{opt} + \varepsilon
\]
Requirements of our smooth function

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

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

$$\text{dist}(f, f_\delta) \leq 2 \cdot \text{opt} + \varepsilon$$
Requirements of our smooth function

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

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

$$f_\delta(x) := \mathbb{E}_{y \sim x}[f(y)]$$
Influence for real-valued functions

Two possible definitions of influence for \( f: \{-1,1\}^n \rightarrow \mathbb{R} \)

**Definition:** The \( L_2 \) influence of \( x_i \) with respect to a function \( f \) is

\[
\inf_{i}^{(2)}(f) := \mathbb{E}_x \left[ \left( f(x) - f(x^{\sim i}) \right)^2 \right]
\]

**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

\[
\inf_{i}^{(1)}(f) := \mathbb{E}_x \left[ \| f(x) - f(x^{\sim i}) \| \right]
\]

**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 $f_\delta$

**Desired property**: For any $f: \{-1,1\}^n \rightarrow \{-1,1\}$ 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. $f_\delta$.

**Fact 1**: For any $f: \{-1,1\}^n \rightarrow \{-1,1\}$ 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 $f: \{-1,1\}^n \rightarrow \mathbb{R}$ 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 $f_\delta$ to be close to $f$  

2. We want to be able to efficiently simulate approximate queries to $f_\delta$ given query access to $f$  

3. We want $f_\delta$ to have few variables of high influence  

Property: For any $f : \{-1,1\}^n \rightarrow \{-1,1\}$ 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. $f_\delta$. 
$f$ opt-close $\epsilon$-close $f_{\delta}$ has few highly influential variables

Putting everything together

$T$ depth-$d$

3opt + $\epsilon$-close

2opt + $\epsilon$-close

5opt + 3$\epsilon$-close

3opt + 2$\epsilon$-close

$\text{FIND}'(f_{\delta}, d, \tau)$
We’ve proven

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

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

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

Can we turn $5 \cdot \text{opt}$ into $1 \cdot \text{opt}$?
More careful analysis: $5 \cdot \text{opt} \to 1 \cdot \text{opt}$
**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* : *inf*(_i_)(*f*) > τ, and *k* = 1, ..., *s* − 1, let

\[
T_{i,k} = \begin{cases} 
\text{FIND}(f_{x_i=-1}, d-1, k) & \text{if } f_{x_i=-1}(d-1,k) > \tau \\
\text{FIND}(f_{x_i=1}, d-1, s-k) & \text{if } f_{x_i=1}(d-1,s-k) > \tau
\end{cases}
\]

Return and cache the \( T_{i,k} \) with minimal error w.r.t. \( f \).
• Bound the total number of calls to $\text{FIND'}$.
• We reach the recursive step at most once per pair $(\pi, k)$
• Number of pairs $= (d/\tau)^d \cdot 2^d = s^{O(\log \log s)}$
• Number of calls per pair $= 2s \cdot |\{i \mid \inf_i(f) > \tau\}|$
• Runtime is still $s^{O(\log \log s)}$
Our result in context

Algorithms for learning DTs

Hypothesis

DT size

- $n^{O(\log s)}$ (relaxed proper)
- $s$ (strictly proper)
- N/A (non-proper)

Time complexity (when $s = \text{poly}(n)$)

- $\text{poly}(n)$
- $n^{O(\log \log n)}$
- $n^{O(\log n)}$