Open Problem: Properly learning decision trees in polynomial time?

Guy Blanc GBLANC@STANFORD.EDU
Stanford

Jane Lange JLANGE@MIT.EDU
MIT

Mingda Qiao MQIAO@STANFORD.EDU
Stanford

Li-Yang Tan LYTAN@STANFORD.EDU
Stanford

Editors: Po-Ling Loh and Maxim Raginsky

Abstract

The authors recently gave an $n^{O(\log\log n)}$ time membership query algorithm for properly learning decision trees under the uniform distribution (Blanc et al., 2021). The previous fastest algorithm for this problem ran in $n^{O(\log n)}$ time, a consequence of Ehrenfeucht and Haussler (1989)'s classic algorithm for the distribution-free setting. In this article we highlight the natural open problem of obtaining a polynomial-time algorithm, discuss possible avenues towards obtaining it, and state intermediate milestones that we believe are of independent interest.

Keywords: Decision trees, proper learning, analysis of Boolean functions

1. Introduction

Decision trees are one of the most intensively studied concept classes in learning theory. In this article we focus on the problem of *properly* learning decision trees, where the learning algorithm is expected to return a hypothesis that is itself a decision tree. Decision tree hypotheses are of particular interest because of their simple structure, and they are the canonical example of a highly interpretable model. Indeed, it is natural to seek decision tree hypotheses even when learning concepts that are not themselves decision trees. Algorithms such as ID3, CART, and C4.5 that construct decision tree representations of datasets are among the most popular and empirically successful algorithms in everyday machine learning practice.

We focus on the setting of PAC learning under the uniform distribution with membership queries, where the learner is given query access to an unknown size-s decision tree target $f : \{0,1\}^n \to \{0,1\}$ and is expected to construct a decision tree hypothesis $h : \{0,1\}^n \to \{0,1\}$ satisfying $\Pr_{\boldsymbol{x}\sim\{0,1\}^n}[f(\boldsymbol{x}) \neq h(\boldsymbol{x})] \leq \varepsilon$, where $\boldsymbol{x} \sim \{0,1\}^n$ is uniform random.

The main open problem of this article is the following:

Open Problem 1 *Design a* $\mathrm{poly}(n, s, 1/\varepsilon)$*-time membership query algorithm for properly learning size-*s *decision trees over* n *variables to error* ε *under the uniform distribution.*

Regarding the use of membership queries, it would of course be preferable if the algorithm does not require them and instead only uses uniform random labeled examples. However, there are two significant barriers to such an algorithm. First, no such statistical query algorithm (SQ) exists: any SQ algorithm for learning size-s decision trees has to make at least $n^{\Omega(\log s)}$ SQs (Blum et al., 1994), and therefore run in at least that much time. Second, since every k-junta is a decision tree of size 2^k, a $\mathrm{poly}(n, s)$ time algorithm for learning size-s decision trees yields a polynomial-time algorithm for leaning $\log(n)$-juntas. Designing such an algorithm that uses only random examples would represent a breakthrough on one of the central and longstanding open problems of learning theory (Blum and Langley, 1997); indeed, it is reasonable to conjecture that there are no polynomial time algorithms for learning k-juntas from random examples for any $k = \omega_n(1)$.

We also mention here the distinction between a *strictly* proper versus *weakly* proper algorithm: the former returns a size-s decision tree hypothesis for a size-s target, whereas the latter can return a decision tree of any size. Open Problem 1 is open even for weakly proper algorithms.

2. Background and current status

Ehrenfeucht and Haussler (1989) were the first to study the problem of properly learning decision trees. They gave an $n^{O(\log s)}$-time weakly proper algorithm that works in the more general distribution-free setting and relies only on random examples. Subsequently, two additional algorithms were designed in the uniform-distribution setting, both of which being substantially different from Ehrenfeucht and Haussler (1989)'s and from each other. First, Mehta and Raghavan (2002) gave an $n^{O(\log s)}$ time strongly proper algorithm. More recently, Blanc et al. (2020) designed a membership query algorithm that runs in $\mathrm{poly}(n) \cdot s^{O(\log s)}$ time. For the standard setting where $s = \mathrm{poly}(n)$, all three algorithms run in quasipolynomial time, $n^{O(\log n)}$.

Recent work of the authors gives a uniform-distribution membership query algorithm that runs in *almost-polynomial* time, $n^{O(\log \log n)}$, bringing us a step closer to the resolution of Open Problem 1. The algorithm is strongly proper and additionally works in the agnostic setting (Haussler, 1992; Kearns et al., 1994):

Theorem 2 (Blanc et al. (2021)) *There is an algorithm which, given as input $\varepsilon > 0$ and $s \in \mathbb{N}$, and query access to $f : \{0,1\}^n \to \{0,1\}$ that is promised to be opt_s-close to a size-s decision tree, runs in time*

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

and outputs a size-s decision tree T that is w.h.p. $(\mathrm{opt}_s + \varepsilon)$-close to f. If f is monotone, the algorithm does not need membership queries and relies only on random examples.

Table 1 summarizes the performance guarantees of the algorithms discussed in this section.

3. Possible approaches

In this section we discuss how the techniques of Blanc et al. (2021) could lead to further progress on Open Problem 1.

Reference	Running time	Hypothesis size	Access to target	Agnostic?
Ehrenfeucht and Haussler (1989)	$n^{O(\log s)}$	$n^{O(\log s)}$	Random examples	×
Mehta and Raghavan (2002)	$n^{O(\log s)}$	s	Random examples	✓
Blanc et al. (2020)	$\mathrm{poly}(n) \cdot s^{O(\log s)}$	$s^{O(\log s)}$	Queries	×
Blanc et al. (2021)	$\mathrm{poly}(n) \cdot s^{O(\log \log s)}$	s	Queries	✓

Table 1: Algorithms for properly learning size-s decision trees. Ehrenfeucht and Haussler (1989)'s algorithm works in the more general distribution-free setting, whereas all others work in the uniform-distribution setting.

Adaptive greediness. The algorithm of Blanc et al. (2020) is a greedy algorithm: At each step, it places the variable with the largest influence, a natural measure of variable importance, as the root. This idea is extended in Blanc et al. (2021): Rather than greedily place down the variable with largest influence, that work proposes a brute force search of the top-k most influential variables to find the best root. They are able to show that, setting $k = \mathrm{polylog}(s)$ and building a tree of depth $\log s$, the algorithm will build a high accuracy tree. The resulting runtime is $k^{O(\log s)} = s^{O(\log \log s)}$.

Perhaps k need not be so big at every iteration? Can algorithms that choose a different amount of greediness (different k) at each step maintain accuracy while improving runtime? For example, it's not hard to show that an algorithm that uses $k = \mathrm{polylog}(s)$ for the first $\frac{\log s}{\log \log s}$ levels, and then $k = O(1)$ for the remaining $\log s - \frac{\log s}{\log \log s}$ would run in poly-time. Is such adaptive greediness sufficient to learn an accurate tree?

New splitting criteria. The algorithm of Mehta and Raghavan (2002) is based on a dynamic programming approach—to build an accurate decision tree for function f, we first solve the subproblems of building trees for restrictions of f to at most $O(\log s)$ variables. A priori, there are $n^{O(\log s)}$ such restrictions, and this is the main bottleneck in the runtime. The algorithm of Blanc et al. (2021) can then be viewed as a refinement of this approach: Their structural result states that for each restricted function f_π, there exists a subset of $\mathrm{polylog}(s) \ll n$ variables that contains a near-optimal choice for the variable to be queried at the root. Furthermore, this subset can be identified by estimating the influence of each variable w.r.t. f_π. This allows us to reduce the number of subproblems to $(\mathrm{polylog}(s))^{O(\log s)} = s^{O(\log \log s)}$.

We note that even *slightly* strengthening this structural result would immediately give a polynomial-time algorithm. Suppose that at each step, the algorithm tries to find the variables to be queried at the first ℓ levels of the tree (rather than only the root). Assuming that the number of such choices is at most N_ℓ, the number of subproblems is at most $(N_\ell \cdot 2^\ell)^{O(\log(s))/\ell}$, which is $\mathrm{poly}(s)$ if $N_\ell = 2^{O(\ell)}$. More concretely, can we pin down $\ell = \Theta(\log \log s)$ levels at once, while exploring only $\mathrm{polylog}(s)$ different choices at each step? If the influence itself is not enough for identifying a small subset of near-optimal choices, would a *slightly* "higher-order" splitting criterion suffice?

4. Intermediate milestones

In this section we state a couple of intermediate milestones towards Open Problem 1 that we believe are of independent interest.

Monotone targets. A function f is *monotone* if for any $x \prec x'$ in the partial order on $\{0,1\}^n$, we have $f(x) \leq f(x')$. Monotone functions have some complexity restrictions that can allow for stronger learning algorithms. For example, in the improper setting, polynomial-time learning algorithms exist for both monotone functions (O'Donnell and Servedio, 2007) and general functions (Kushilevitz and Mansour, 1993), but the learner for general functions requires membership queries whereas the learner for monotone functions requires only random examples. A promising and independently interesting intermediate step towards Open Problem 1 is to design a polynomial-time proper learner under the assumption that the target function is monotone; potentially even one that uses only random examples.

More expressive hypotheses. An algorithm for properly learning decision trees returns a decision tree hypothesis. As an intermediate step towards such an algorithm, one could consider allowing the algorithm return more expressive hypotheses that nevertheless share the desirability of decision trees (e.g. interpretability). Two concrete examples are branching programs—generalizations of decision trees where the underlying graph is a DAG—and generalized decision trees whose internal nodes branch on the outcomes of halfspaces instead of singleton variables. Both these representations are easily seen to be exponentially more succinct than decision trees. In the context of boosting, branching program hypotheses have been shown to be enable exponential improvements over the decision tree hypotheses (Mansour and McAllester, 2002; Long and Servedio, 2005).

Acknowledgements

Guy and Li-Yang are supported by NSF CAREER Award 1942123. Mingda is supported by DOE Award DE-SC0019205 and ONR Young Investigator Award N00014-18-1-2295. Jane is supported by NSF Award CCF-2006664.

References

Guy Blanc, Jane Lange, and Li-Yang Tan. Top-down induction of decision trees: rigorous guarantees and inherent limitations. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 1–44, 2020.

Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Properly learning decision trees in almost polynomial time. In *Proceedings of the 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 920–929, 2021.

Avirm Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 253–262, 1994.

Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and computation*, 100(1):78–150, 1992.

Michael Kearns, Robert Schapire, and Linda Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2/3):115–141, 1994.

Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, December 1993.

Philip M Long and Rocco A Servedio. Martingale boosting. In *International Conference on Computational Learning Theory*, pages 79–94. Springer, 2005.

Yishay Mansour and David McAllester. Boosting using branching programs. *Journal of Computer and System Sciences*, 64(1):103–112, 2002.

Dinesh Mehta and Vijay Raghavan. Decision tree approximations of boolean functions. *Theoretical Computer Science*, 270(1-2):609–623, 2002.

Ryan O'Donnell and Rocco Servedio. Learning monotone decision trees in polynomial time. *SIAM Journal on Computing*, 37(3):827–844, 2007.