Encyclopedia of Algorithms
DOI 10.1007/978-3-642-27848-8_193-2

Learning with the Aid of an Oracle (1996; Bshouty, Cleve, Gavaldà, Kannan, Tamon)

Christino Tamon*
Department of Mathematics and Computer Science, Clarkson University, Potsdam, NY, USA Q1

Index Terms: Exact learning via queries; Boolean circuits; Disjunctive normal form

Problem Definition

In the exact learning model of Angluin [2], a learning algorithm A must discover an unknown function $\mathsf{f} : \{0,1\}^n \to \{0,1\}$ that is a member of a known class C of Boolean functions. The learning algorithm can make at least one of the following types of queries about f:

- Equivalence query $\mathsf{EQ_f(g)}$, for a candidate function g:
  The reply is either "yes," if $\mathsf{g} \Leftrightarrow \mathsf{f}$, or a counterexample a with $\mathsf{g}(a) \neq \mathsf{f}(a)$, otherwise.
- Membership query $\mathsf{MQ_f}(a)$, for some $a \in \{0,1\}^n$:
  The reply is the Boolean value $\mathsf{f}(a)$.
- Subset query $\mathsf{SubQ_f(g)}$, for a candidate function g:
  The reply is "yes," if $\mathsf{g} \Rightarrow \mathsf{f}$, or a counterexample a with $\mathsf{f}(a) < \mathsf{g}(a)$, otherwise.
- Superset query $\mathsf{SupQ_f(g)}$, for a candidate function g:
  The reply is "yes," if $\mathsf{f} \Rightarrow \mathsf{g}$, or a counterexample a with $\mathsf{g}(a) < \mathsf{f}(a)$, otherwise.

A disjunctive normal formula (DNF) is a depth-2 OR-AND circuit whose size is given by the number of its AND gates. Likewise, a conjunctive normal formula (CNF) is a depth-2 AND-OR circuit whose size is given by the number of its OR gates. Any Boolean function can be represented as both a DNF or a CNF formula. A k-DNF is a DNF where each AND gate has a fan-in of at most k; similarly, we may define a k-CNF.

Problem For a given class C of Boolean functions, such as polynomial-size Boolean circuits or disjunctive normal form (DNF) formulas, the goal is to design polynomial-time learning algorithms for any unknown $\mathsf{f} \in \mathsf{C}$ and ask a polynomial number of queries. The output of the learning algorithm should be a function g of polynomial size satisfying $g \Leftrightarrow f$. The polynomial functions bounding the running time, query complexity, and output size are defined in terms of the number of inputs n and the size of the smallest representation (Boolean circuit or DNF) of the unknown function f.

Key Results

One of the main results proved in [5] is that Boolean circuits and disjunctive normal formulas are exactly learnable using equivalence queries and access to an NP oracle.

*E-mail: tino@clarkson.edu

DOI 10.1007/978-3-642-27848-8_193-2

Theorem 1. *The following tasks can be accomplished with probabilistic polynomial-time algorithms that have access to an* NP *oracle and make polynomially many equivalence queries:*

- *Learning* DNF *formulas of size s using equivalence queries that are depth-*3 AND-OR-AND *formulas of size $O(sn^2/\log^2 n)$.*
- *Learning Boolean circuits of size s using equivalence queries that are circuits of size $O(sn + n\log n)$.*

The idea behind this result is simple. Any class C of Boolean functions is exactly learnable with equivalence queries using the Halving algorithm of Littlestone [11]. This algorithm asks equivalence queries that are the *majority* of candidate functions from C. These are functions in C that are consistent with the counterexamples obtained so far by the learning algorithm. Since each such majority query eliminates at least half of the candidate functions, $\log_2 |\mathsf{C}|$ equivalence queries are sufficient to learn any function in C. A problem with using the Halving algorithm here is that the majority query has exponential size. But, it can be shown that a majority of a polynomial number of uniformly random candidate functions is a good enough approximator to the majority of all candidate functions. Moreover, with access to an NP oracle, there is a randomized polynomial time algorithm for generating random uniform candidate functions due to Jerrum, Valiant, and Vazirani [7]. This yields the result.

The next observation is that subset and superset queries are apparently powerful enough to simulate both equivalence queries and the NP oracle. This is easy to see since the tautology test $\mathsf{g} \Leftrightarrow 1$ is equivalent to $\mathsf{SubQ_f}(\overline{\mathsf{g}}) \wedge \mathsf{SubQ_f}(\mathsf{g})$, for any unknown function f; and, $\mathsf{EQ_f}(\mathsf{g})$ is equivalent to $\mathsf{SubQ_f}(\mathsf{g}) \wedge \mathsf{SupQ_f}(\mathsf{g})$. Thus, the following generalization of Theorem 1 is obtained.

Theorem 2. *The following tasks can be accomplished with probabilistic polynomial-time algorithms that make polynomially many subset and superset queries:*

- *Learning* DNF *formulas of size s using equivalence queries that are depth-*3 AND-OR-AND *formulas of size $O(sn^2/\log^2 n)$.*
- *Learning Boolean circuits of size s using equivalence queries that are circuits of size $O(sn + n\log n)$.*

Stronger deterministic results are obtained by allowing more powerful complexity-theoretic oracles. The first of these results employ techniques developed by Sipser and Stockmeyer [12, 13].

Theorem 3. *The following tasks can be accomplished with deterministic polynomial-time algorithms that have access to an Σ_3^p oracle and make polynomially many equivalence queries:*

- *Learning* DNF *formulas of size s using equivalence queries that are depth-*3 AND-OR-AND *formulas of size $O(sn^2/\log^2 n)$.*
- *Learning Boolean circuits of size s using equivalence queries that are circuits of size $O(sn + n\log n)$.*

In the following result, C is an infinite class of functions containing functions of the form $\mathsf{f} : \{0,1\}^\star \to \{0,1\}$. The class C is p-evaluatable if the following tasks can be performed in polynomial time:

DOI 10.1007/978-3-642-27848-8_193-2

- Given y, is y a valid representation for any function $\mathsf{f}_y \in \mathsf{C}$?
- Given a valid representation y and $x \in \{0,1\}^\star$, is $\mathsf{f}_y(x) = 1$?

Theorem 4. *Let* C *be any p-evaluatable class. The following statements are equivalent:*

- C *is learnable from polynomially many equivalence queries of polynomial size (and unlimited computational power).*
- C *is learnable in deterministic polynomial time with equivalence queries and access to a* Σ_5^p *oracle.*

For exact learning with membership queries, the following results are proved.

Theorem 5. *The following tasks can be accomplished with deterministic polynomial-time algorithms that have access to an* NP *oracle and make polynomially many membership queries (in* n, DNF *and* CNF *sizes of* f*, where* f *is the unknown function):*

- *Learning monotone Boolean functions.*
- *Learning* $O(\log n)$-$\mathsf{CNF} \bigcap O(\log n)$-$\mathsf{DNF}$.

The ideas behind the above result use techniques from [2,4]. For a monotone Boolean function f, the standard closure algorithm uses both equivalence and membership queries to learn f using candidate functions g satisfying $\mathsf{g} \Rightarrow \mathsf{f}$. The need for membership can be removed using the following observation. Viewing $\neg\mathsf{f}$ as a monotone function on the inverted lattice, we can learn f and $\neg\mathsf{f}$ simultaneously using candidate functions g, h, respectively, that satisfy $\mathsf{g} \Rightarrow \mathsf{h}$. The NP oracle is used to obtain an example a that either helps in learning f or in learning $\neg\mathsf{f}$; when no such example can be found, we have learned f.

Theorem 6. *Any class* C *of Boolean functions that is exactly learnable using a polynomial number of membership queries (and unlimited computational power) is exactly learnable in expected polynomial time using a polynomial number of membership queries and access to an* NP *oracle.*

Moreover, any p-evaluatable class C *that is exactly learnable from a polynomial number of membership queries (and unlimited computational power) is also learnable in deterministic polynomial time using a polynomial number of membership queries and access to a* Σ_5^p *oracle.*

Theorems 4 and 6 showed that information-theoretic learnability using equivalence and membership queries can be transformed into computational learnability at the expense of using the Σ_5^p and NP oracles, respectively.

Applications

The learning algorithm for Boolean circuits using equivalence queries and access to an NP oracle has found an application in complexity theory. Watanabe (see [10]) showed an improvement on a known theorem of Karp and Lipton [8]: if NP has polynomial-size circuits, then the polynomial-time hierarchy PH collapses to $\mathsf{ZPP}^{\mathsf{NP}}$. Subsequently, Aaronson (see [1]) showed that queries to

DOI 10.1007/978-3-642-27848-8_193-2

the NP oracle used in the learning algorithm (for Boolean circuits) cannot be parallelized by any relativizing techniques.

Some techniques developed in Theorem 5 for exact learning using membership queries of monotone Boolean functions have found applications in data mining [6].

Open Problems

It is unknown if there are polynomial-time learning algorithms for Boolean circuits and DNF formulas using equivalence queries (without complexity-theoretic oracles). There are strong cryptographic evidence that Boolean circuits are not learnable in polynomial-time (see [3] and the references therein). The best running time for learning DNF formulas is $2^{\tilde{O}(n^{1/3})}$ as given by Klivans and Servedio [9]. It is unclear if membership queries help in this case.

Experimental Results

None reported.

Data Sets

None reported.

URL to Code

None reported.

Cross References

For related learning results, see

Q2 ▸Learning DNF Formulas (1997; Jackson)

Q3 ▸Learning Automata (2000; Beimel, Bergadano, Bshouty, Kushilevitz, and Varricchio)

in this encyclopedia.

Recommended Reading

Q4

1. Aaronson S (2006) Oracles are subtle but not malicious. In: Proceedings of the 21st annual IEEE conference on computational complexity (CCC'06), Prague, pp 340–354
2. Angluin D (1988) Queries and concept learning. Mach Learn 2:319–342
3. Angluin D, Kharitonov M (1995) When Won't Membership Queries Help? J Comput Syst Sci 50:336–355

DOI 10.1007/978-3-642-27848-8_193-2

4. Bshouty NH (1995) Exact learning boolean function via the monotone theory. Inf Comput 123:146–153
5. Bshouty NH, Cleve R, Gavaldà R, Kannan S, Tamon C (1996) Oracles and queries that are sufficient for exact learning. J Comput Syst Sci 52(3):421–433
6. Gunopolous D, Khardon R, Mannila H, Saluja S, Toivonen H, Sharma RS (2003) Discovering all most specific sentences. ACM Trans Database Syst 28:140–174
7. Jerrum MR, Valiant LG, Vazirani VV (1986) Random generation of combinatorial structures from a uniform distribution. Theor Comput Sci 43:169–188
8. Karp RM, Lipton RJ (1980) Some connections between nonuniform and uniform complexity classes. In: Proceedings of the 12th annual ACM symposium on theory of computing, Los Angeles, pp 302–309
9. Klivans AR, Servedio RA (2004) Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. J Comput Syst Sci 68:303–318
10. Köbler J, Watanabe O (1998) New collapse consequences of np having small circuits. SIAM J Comput 28:311–324
11. Littlestone N Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. Mach Learn 2:285–318 (1987)
12. Sipser M (1983) A complexity theoretic approach to randomness. In: Proceedings of the 15th annual ACM symposium on theory of computing, Boston, pp 330–334
13. Stockmeyer LJ (1985) On approximation algorithms for $\#P$. SIAM J Comput 14:849–861