The Complexity of Learning with Queries

Ricard Gavaldà*
Department of Software (LSI)
Universitat Politècnica de Catalunya
Pau Gargallo 5
E-08028 Barcelona, Spain
E-mail: gavalda@lsi.upc.es

Revised version, September 27th, 1994

Abstract

We survey recent research concerning the qualitative complexity of Angluin's model of learning with queries. In this model, there is a learner that tries to identify a target concept by means of queries to a teacher. Thus, the process can be naturally formulated as an oracle computation. Among the results we review there are: characterizations of the power of different learning protocols by complexity classes of oracle machines; relations between the complexity of learning and the complexity of computing advice functions for nonuniform classes; and combinatorial characterizations of the concept classes that are learnable in specific protocols.

1 Introduction

This paper is a survey on recent results and ideas on the qualitative complexity of Angluin's model of learning with queries, also known as query learning

*This research was supported in part by the ESPRIT Basic Research Actions Program of the EC, projects ALCOM and NEUROCOLT, and by DGICYT, project number PB92–0709. A preliminary version of this report was presented at the 9th IEEE Conference on Structure in Complexity Theory (Amsterdam, june–july 1994).

or exact learning. This model is quickly becoming one of the most popular ones the computational learning community, receiving almost as much attention as more traditional models such as PAC or inductive inference.

The paper does not discuss at all for which specific problems efficient query learning algorithms have been found; the recent surveys [4, 5] are quite exhaustive on this respect. Also, we discuss only marginally whether query learning is a reasonable model of real-life learning processes, and to which extent it can be used to guide practical implementation of learning systems. Rather, the emphasis is in how concepts and techniques borrowed from structural complexity theory can be used to gain insight on query learning problems. No proofs are given, except for a few previously unpublished results.

The paper is organized as follows. After fixing some basic notation on complexity theory, Section 3 presents the original model of query learning and several variants, such as bounded learning. In Section 4 we present a structural approach to the complexity of learning, proposed by Watanabe [30] and extended by Watanabe and Gavaldà [32]. In particular, we explain how to characterize the computational power of several protocols using restricted subclasses of oracle Δ_2^p machines, and how these characterizations tie the complexity of learning to recent research on the structure of P/poly. In Sections 5 and 6 we present some progress on clarifying the different reasons why learning problems can be hard; in some cases, it may be due to purely information-theoretical reasons, that is, many queries are needed no matter how cleverly asked; in others, it is a complexity-theoretic barrier (such as $P \neq NP$) that prevents efficient learning.

For comparison purposes, we mention occasionally Valiant's model of probably approximately correct (PAC) learning [28], but we do not define it here. For general presentations of computational learning theory, see the books by Natarajan [23] and Anthony and Biggs [7], or the surveys by Angluin [4, 5], Laird [21], Pitt [24], and Warmuth [29].

2 Notation

Most of our notation for complexity theory is standard and can be found in textbooks such as [8]. We use a fixed alphabet Σ with more than one symbol; we can assume $\Sigma = \{0, 1\}$, knowing that larger alphabets can be encoded into this one if necessary. For $x \in \Sigma^\star$, $|x|$ is the length of x. For a language $A \subseteq \Sigma^\star$, $A^{=n}$ and $A^{\leq n}$ denote the sets of strings with length equal to n and at most n, respectively.

For a finite set X, $\|X\|$ is the cardinality of X.

We use often classes of Turing machines defining complexity classes. For example, PF is the class of functions computable in polynomial time, so "the class of PF machines" means all polynomial-time transducers. To denote the machine class relativized to X we write for example PF(X); here, X can be an oracle, a class of oracles, a machine, or another class of machines. When we want to denote the corresponding class of oracle machines, but leave the oracle unspecified, we write simply PF(). Thus, for example, PF(NP()) means the class of PF machines each querying some NP machine that in turn can ask another oracle.

The only non-standard complexity classes that we use are three subclasses of NP() machines.

- $\mathrm{NP}_1()$ is the set of nondeterministic polynomial-time machines that make at most one query to their oracle on each computation path.

- $\mathrm{NP}_m()$ is the subclass of $\mathrm{NP}_1()$ where each computation path making a query accepts if and only if the query receives a 'yes' answer from the oracle. These machines compute nondeterministic polynomial-time many-one reductions as defined in [20].

- $\mathrm{NP}_{\overline{m}}()$ is the subclass of $\mathrm{NP}_1()$ where each computation path making a query accepts if and only if the query receives a 'no' answer from the oracle.

We also use the machines defining the Σ and Δ levels of the polynomial hierarchy (PH). These are defined inductively by: $\Sigma_1^{\mathrm{p}} = \mathrm{NP}$, $\Delta_1^{\mathrm{p}} = \mathrm{PF}$, and, for $k \geq 2$, $\Sigma_k^{\mathrm{p}} = \mathrm{NP}(\Sigma_{k-1}^{\mathrm{p}})$ and $\Delta_k^{\mathrm{p}} = \mathrm{PF}(\Sigma_{k-1}^{\mathrm{p}})$. Note that we use functional versions of the Δ levels, not the decisional ones.

3 Learning with Queries: Definitions

We deal only with language learning, i.e., each concept to learn is a language over the alphabet Σ. The definitions that follow are essentially those in [24, 29, 30]. The notation is mostly from [30, 31]. More general definitions are possible; for example, to deal with learning of functions that are not 0/1-valued.

Representation Classes. We use *representation classes* to specify learning problems. The rationale behind the following definition is that the difficulty of learning the same class of concepts (or languages) can vary enormously depending on the syntax of the output we have to give. The problem is not correctly specified until we fix a description language for concepts.

Definition 1 *A representation class C is a pair (R, Φ), where*

- *$R \subseteq \Sigma^\star$ is the set of representations.*
- *$\Phi : \Sigma^\star \to 2^{\Sigma^\star}$ is the semantic function.*

For each $r \in R$, the language $\Phi(r)$ is the concept represented by r, and $\Phi(R)$ is the set of all concepts that can be represented in R.

If $c \in \Phi(R)$, by the size of concept c we mean the length of a shortest element in R that represents c.

Some of the representation classes that we will use in this paper are the following.

- CIR $= (R_{\mathrm{CIR}}, \Phi_{\mathrm{CIR}})$ represents boolean circuits: R_{CIR} is the set of strings encoding, in some straightforward way, boolean circuits with one output gate; Φ_{CIR} maps each encoding to the finite set of binary strings on which the circuit evaluates to 1. (Note: there is no notion of "polynomial size" for circuits implied here.)
- DFA $= (R_{\mathrm{DFA}}, \Phi_{\mathrm{DFA}})$ stands for deterministic finite automata. R_{DFA} is the set of encodings of dfa acceptors, and Φ_{DFA} maps each encoding to the corresponding regular language.
- Now it should be clear how to define classes such as NFA (nondeterministic finite automata), CFG (context-free grammars), CNF (boolean formulas in conjunctive normal form), and DNF (boolean formulas in disjunctive normal form).

We say that a representation class is *fixed-length* if every concept in the class contains only strings of a single length. Thus, CIR is fixed-length but DFA is not.

Observe that concepts in the classes above are finite, regular, or context-free languages, hence very simple from the point of view of complexity theory. Also, the encoding of concepts into strings in these classes is only of moderate complexity: evaluation of circuits, finite automata, formulas, or grammars on given inputs is also solvable in polynomial time. This is the case in the vast majority of computational learning research: The learning agent has access to a simple oracle, rarely beyond P – the problem is, of course, that it does not know which one.

Many times we will restrict ourselves to classes with these nice properties. The following definition has been used often, more or less explicitly and with different names.

Definition 2 *A representation class (R, Φ) is polynomially honest (or honest in short) if it satisfies:*

1. *$R \in \mathrm{P}$, i.e., we can check the syntax of representations in polynomial time.*
2. *$\{ \ (r, x) \ \mid x \in \Phi(r) \ \} \in \mathrm{P}$, i.e., not only each concept is in P but there is a uniform way to check membership to concepts in polynomial time.*

Certainly there are interesting learning problems that do not satisfy these conditions. Think, for example, on trying to learn boolean formulas with some quantified variables, a class whose membership problem is PSPACE-complete. A risk in these cases is that learners "cheat", querying their complex oracle not only to identify it but also to boost their computation power.

Queries. A *teacher* is a function mapping *query strings* to *answer strings*. Learners want to use teachers that provide consistent information about the *target concept* they are trying to identify. The meaning of "consistency" here is what defines the different query types.

The query types we mention in this paper are taken from [2]. The easiest one is a *Membership query*, in which the teacher only replies 'yes' or 'no' according to whether the argument to the query is or is not in the target

Membership (Mem)
Input: $x \in \Sigma^\star$
Output: 'yes' if $x \in target$, 'no' otherwise.

Equivalence (Equ)
Input: $r \in R$
Output: 'yes' if $\Phi(r) = target$, $w \in \Phi(r) \ \Delta \ target$ otherwise.

Subset (Sub)
Input: $r \in R$
Output: 'yes' if $\Phi(r) \subseteq target$, $w \in \Phi(r) - target$ otherwise.

Superset (Sup)
Input: $r \in R$
Output: 'yes' if $\Phi(r) \supseteq target$, $w \in target - \Phi(r)$ otherwise.

Table 1. Query types

concept. For *Equivalence queries,* the input is a representation r and the output is either 'yes', if r represents exactly the target, or otherwise a *counterexample string* w that witnesses the difference. Other query types we use are Subset and Superset. Their definitions are in Table 1 together with the shorthands we use for each query type.

By a *protocol* we mean a set of these query types. A *teacher for a concept* c under a protocol α is one that answers all queries in α consistently with respect to c (assume, for example, that different types of queries are prefixed by special marks telling them apart). Observe that for Equivalence, Subset, and Superset queries, more than one counterexample can exist to a given query. Hence, more than one consistent teacher for every concept can exist.

When designing a learning algorithm, how reasonable is it to assume that these queries are indeed available? In many situations, Membership queries are probably feasible, be it by querying a human expert, a database, or a program simulation.

On the other hand, answering exactly Equivalence queries seems hopeless in most domains, even if the teacher already has some representation for the target: the problem is NP-hard for circuits or formulas, and plainly undecidable for context-free grammars. Still, the protocol formed by Membership and Equivalence queries is widely used in query learning research. The reason is that algorithms using Equivalence queries can be transformed into others using more feasible types of queries, such as PAC-algorithms with a source of random examples (as explained in [1, 2]), or mistake-bounded on-line algorithms, that receive a sequence of examples to predict and try to minimize the number of prediction errors [22].

The combination of Subset and Superset queries seems even more difficult. Not only can it simulate Equivalence queries; as observed recently by Bshouty *et. al.* [11], one Subset and one Superset query to any teacher for circuits or formulas allow to solve any NP-complete predicate. Indeed, observe that a circuit C is constantly 0 if and only if $\Phi(C) \subseteq \mathit{target}$ and $\mathit{target} \subseteq \Phi(\neg C)$, no matter what *target* is.

Learning Systems. A *learner* is a computing device that can query a teacher as a functional oracle. To be specific, think of a learner as a multitape oracle Turing machine, so that interacting with the teacher once means writing down the query on an oracle tape, getting the answer on an answer tape in unit time, then possibly reading it symbol by symbol.

The learner may or may not have input depending on precise model of learning, as discussed next.

Borrowing the usual notation for oracles, we write $S^T(x)$ to denote the computation of learner S started on input x and querying teacher T. If S has no input, we write simply S^T.

Exact Learning. This is the model proposed by Angluin in the initial papers [1, 2], as well as the one most often used in the literature. Roughly speaking, the goal of the learner here is to interact with the teacher and come up with a representation for *exactly* the target concept. The learner has no prior information about the target concept, besides the promise that it is chosen from a known representation class. In particular, it has no input.

Since we want to study time-efficient learning, we have to fix a way to measure the running time of the algorithm. The standard convention (as a function of the input length) will not work, simply because a learning algorithm has no input. The solution comes out of the following two observations: 1) It makes sense to give more time to the algorithm to learn more complex concepts; usually, we take the size of (the smallest representation of) the target concept as a measure of its complexity. 2) We are assuming that reading an answer from the teacher takes time linear in its length; this way, an uncooperative teacher can force large running times by choosing long counterexamples even if short ones exist; furthermore, even the kindest teacher may be forced to give a long counterexample if the learner's hypothesis already agrees with the target in all smaller lengths. This suggests to take into account the length of counterexamples when computing the running time.

Definition 3 *A learner S is an exact learner for class $C = (R, \Phi)$ if there is some polynomial p such that for every $r \in R$ and for every teacher T for $\Phi(r)$,*

1. *S^T halts, and the time used in any moment of the computation is at most $p(|r|, m)$, where m is the length of the longest answer received from T so far, and*
2. *S^T outputs some $r' \in R$ such that $\Phi(r) = \Phi(r')$.*

Of course, this definition assumes that S and T agree to use a common protocol, or set of queries. To make explicit that the protocol is α, we say that C is α-learnable and that S is an α-learner.

Note that this is a worst-case definition; S must learn exactly and quickly from *any* teacher consistent with the target.

Bounded Learning. Watanabe [30, 31] proposed an apparently weaker definition of learning, which is inspired by the following observation. After learning a concept, most often we want to use the resulting representation to predict future instances. In many practical situations, we are not interested in instances of arbitrary lengths. We will be happy if our representation classifies correctly all instances up to some length bound, that we can estimate beforehand.

If we do not care about very long instances, it seems unfair to require a teacher that answers correctly about every string.

Definition 4 *For $m \geq 0$, an m-bounded teacher T for a concept c answers a Membership query x with 'yes' if $|x| \leq m$ and $x \in c$, with 'no' if $|x| \leq m$ and $x \notin c$ (and the answer is not reliable if $|x| > m$). Similarly, for an equivalence query $r \in R$, it returns either some $x \in \Phi(r)^{\leq m} \,\Delta\, c^{\leq m}$, or else 'yes' if no such x exists (i.e., $\Phi(r)^{\leq m} = c^{\leq m}$). Bounded teachers for the other query types are defined analogously.*

In Watanabe's notion of bounded learning, we want to obtain a representation that is correct up to length m when using a teacher that is $q(m)$-bounded, where q is a polynomial. (Using $q(m)$- instead of m-bounded teachers is of technical help in some of the coming theorems, as to identify a concept up to length m we may need to ask queries somewhat longer than m). The *length bound* m is given as input to the learner.

Many times, there is some additional information known in advance about the target concept, other than its membership to a concept class. For example, most often we have an idea of how complex it is, to the point that we can make a rough guess about the size of its representations. Watanabe's definition assumes that the learner is given a *size bound* n, some upper bound on the size of the target concept. The algorithm may use this information, for example, to set up initial data structures of the appropriate size. Then it makes sense to measure the running time of the algorithm relative to n and not to the actual size of the target concept.

All in all, bounded learning is defined as follows.

Definition 5 *A learner S is a bounded learner for class $C = (R, \Phi)$ if there are polynomials p and q such that for every $n, m \geq 0$, for every $r \in R$ such that $|r| \leq n$, and for every $q(m)$-bounded teacher T for $\Phi(r)$,*

1. *$S^T(n, m)$ halts in time at most $p(n, m)$, and*
2. *$S^T(n, m)$ outputs some $r' \in R$ such that $\Phi(r)^{\leq m} = \Phi(r')^{\leq m}$.*

Note that, in case the teacher is not $q(m)$-bounded, or the target concept has no representation of size n or less, $S^T(n, m)$ may not halt, or halt without output, or give an r' with no relation whatsoever to the target.

When we say "polynomial time" referring to a learning task, we will mean condition (1) in Definitions 3 or 5, depending on whether learning is exact or bounded.

Proper Learning. Usually, we consider learners that try to learn a class $C = (R, \Phi)$ from teachers for the same class, that is, the learners limit their Equivalence, Subset, and Superset queries to strings in R. If C is learnable in this way we say that it is *properly learnable.*

Sometimes, the class C itself has too little expressive power to achieve learning. In this case, C may be still learnable if we allow queries to a larger class $C' = (R', \Phi')$, where $\Phi(R) \subseteq \Phi'(R')$, and use teachers for C'. Consequently, we should allow the algorithm to output representations in R'. Only the promise that the target is in $\Phi(R)$ remains. We say in this case that C *is learnable from* C' or *using hypotheses in* C'.

In summary: in this paper, for each learner we should make clear whether it is exact or bounded, whether it is proper or it makes queries from a larger class, and which protocol it uses. By default, we will mean exact and proper learning.

Identification. We need still another notion. Sometimes we want to focus only in the amount of information that has to be communicated between learner and teacher in order to learn, and not on the computing power needed to process it. In this case, we drop the requirement that the learner is a polynomial-time algorithm and we consider (possibly non-computable) strategies, functions telling which is the next query to make given the previous query/answer history.

We say that a class C *can be identified* in a given protocol (as opposed to *learned*) if there is any strategy that achieves condition (2) in the definitions of learning (Definitions 3 and 5).

Condition (1) is replaced in this case by the requirement to make only a polynomial number of queries, each of polynomial size. "Polynomial" here means polynomial in $|r|$ and m in the exact case, or n and m in the bounded case. A negative result for identification means an information-theoretic impossibility, regardless of any computational complexity considerations.

4 Learning as Oracle Computation

We have now formulated learning as a computation relative to an oracle. It is natural to ask whether our knowledge on oracle complexity classes can give some insight on the complexity of learning. However, the two paradigms are sufficiently different so that comparisons are not trivially established.

For example, while learning machines specialize in computing representations for the oracle they are using, relativized complexity classes normally include all sorts of other functions.

Also, usually one asks queries *to* a *known* oracle set, in order to compute a function in a relativized complexity class. Here, we ask queries *about* an *unknown* oracle set, in order to identify it; hence, our algorithm cannot be so much taylored to a particular oracle.[1]

Furthermore, in structural complexity we prefer complexity classes as robust as possible. For example, for an oracle A we like to study the class PF(NP(A)), and not care too much about the specific NP machine that is used on top of the PF one. The point in query learning is that we can only query the oracle in a set of fixed and simple ways. For example, a bounded Equivalence query to a teacher for c is in fact the query

$$\exists\ w\ (\ |w| \leq m \quad \wedge \quad w \in \Phi(r)\ \Delta\ c\ )$$

which is an NP(c) query with input $(r, 0^m)$. Fixing the protocol to be {Equ} is, roughly speaking, agreeing to use only one type of NP(c) query to c, one that in general is not complete. This makes difficult any comparison with standard complexity classes.

To overcome these difficulties, Watanabe [30] proposed to study how machines and learners simulate each other. This section reviews the results of this study, initiated already in [30], and later extended by Watanabe and Gavaldà [32]. Their research focused only on bounded learning. (The reader should have in mind that here we are somewhat simplifying on [30, 32]. Actual results there are often stronger or proved with respect to more general definitions.)

The first task is to restrict oracle machines in a class to behave almost like learners, with respect to the kind of functions they compute. We want machines that find representations of their oracles.

Definition 6 *Let M be an oracle machine and $C = (R, \Phi)$ be a representation class. We say that M finds bounded representations for C if for every $n, m \geq 0$, for every $r \in R$ with $|r| \leq n$, M with input $(0^n, 0^m)$ and relative to oracle $\Phi(r)$ finds $r' \in R$ such that*

$$\Phi(r)^{\leq m} = \Phi(r')^{\leq m}$$

[1]Of course, one can design learning algorithms that have access to a fixed and known oracle set such as SAT, to see how much it helps in the learning process. The distinction between asking queries *about* an unknown oracle and *to* a known oracle has been quite well studied in Gold's model of inference. See [13] for the first case and [12] for the second.

Note that, unlike learners, machines here get their input $(0^n, 0^m)$ in tally format so that the standard notion of polynomial time applies.

Now we can state when a machine class is an upper bound or a lower bound. Informally, machines in an upper bound for a protocol can solve all learning tasks solvable in that protocol. Proving an upper bound will usually be showing how to simulate each query in the protocol using the machine class.

Similarly, if a machine class $\mathcal{M}$ is a lower bound for α, learners can use the queries in α plus some polynomial-time computation to simulate any machine in $\mathcal{M}$, provided this machine is used to compute representations for its oracle. Thus, machines strictly less powerful than those in $\mathcal{M}$ should not be able to simulate every α-learner, at least when restricted to compute representations. This justifies the name of lower bound.

Definition 7 *Let $\mathcal{M}$ be a machine type, α a protocol, and C a representation class. We say that $\mathcal{M}$ is an upper bound for α relative to C if the following (1) implies (2). We say that $\mathcal{M}$ is a lower bound for α relative to C if (2) implies (1).*

1. *C is bounded α-learnable.*
2. *There exists a machine of type $\mathcal{M}$ that finds bounded representations for C.*

It is possible to prove very general upper bounds relying only on the honesty condition.

Theorem 8 *[32] Relative to any honest representation class C,*

1. PF() *is an upper bound for protocol* {Mem}*,*
2. $\mathrm{PF}(\mathrm{NP}_m())$ *is an upper bound for protocol* {Mem,Sup}*,*
3. $\mathrm{PF}(\mathrm{NP}_{\overline{m}}())$ *is an upper bound for protocol* {Mem,Sub}*, and*
4. $\mathrm{PF}(\mathrm{NP}_1())$ *is an upper bound for the protocol containing all queries in Table 1 (and in fact all queries proposed in [2]).*

Hence, the computational power of our protocols lies always within $\Delta_2^{\mathrm{p}}() = \mathrm{PF}(\mathrm{NP}())$. For representation classes with very low expressive power, less power probably suffices. However, we can show that for *some* representation class these bounds are tight – this much power is needed in general for the simulation.

We find such a class by imposing additional structure on a class that is already powerful, such as circuits. Define representation class REPCIR ("repetitive circuits") as the pair $(R_{\mathrm{REPCIR}}, \Phi_{\mathrm{REPCIR}})$ where

- $R_{\mathrm{REPCIR}} = R_{\mathrm{CIR}}$, that is, encodings of 0/1-valued circuits.
- $\Phi_{\mathrm{REPCIR}}(c)$ is the set $(\Phi_{\mathrm{CIR}}(c)\#)^{\star}$, where $\#$ is an additional separator symbol. That is, a Membership query to REPCIR is the conjunction of several Membership queries to CIR.

Using this internal structure, one can show that Subset and Superset queries to REPCIR suffice to simulate general $\mathrm{NP}_m()$ or $\mathrm{NP}_{\overline{m}}()$ computations, when their oracles are also in REPCIR.

Theorem 9 *[32] Relative to representation class* REPCIR*,*

1. PF() *is a lower bound for protocol* {Mem}*,*
2. $\mathrm{PF}(\mathrm{NP}_m())$ *is a lower bound for protocol* {Sup}*,*
3. $\mathrm{PF}(\mathrm{NP}_{\overline{m}}())$ *is a lower bound for protocol* {Sub}*, and*
4. $\mathrm{PF}(\mathrm{NP}_1())$ *is an lower bound for protocol* {Sup,Sub}*.*

(The proofs are not too particular to circuits. One can define repetitive versions of NFA and CFG and some of the lower bound results still hold [32].)

Now, let us point out some connections with purely structural research on the class P/poly.

Recall that the nonuniform class P/poly was defined in [16] as the class of sets having polynomial-length advice functions that can be interpreted in P. The class P/poly is known to be equal to the class of sets having non-uniform polynomial-size circuits (Meyer, in [9]), and from the proof of this fact one can see that we can restrict the original definition of P/poly to advice functions whose outputs are boolean circuits. Thus, for the purposes of this paper, we define advice functions as follows.

Definition 10 *An advice function for a set A is a function $f : 0^\star \to \Sigma^\star$ such that, for every n, $f(0^n)$ encodes a circuit accepting exactly $A^{=n}$.*

The problem then becomes:

> For a relativizable complexity class $\mathcal{CF}$, is it true that $\mathcal{CF}(A)$ contains an advice function for each A in P/poly?

In the work of Ko and Schöning [18] and Schöning [26], it is implicitly shown that this is true for $\mathcal{CF}() = \mathrm{PF}(\Sigma_2^\mathrm{p}())$. An incomparable bound was shown later in [14], which was further improved by Köbler [19] to give the best result known up to date:

Theorem 11 *[19] Every set A in* P/poly *has some advice function in* $\mathrm{PF}(\mathrm{NP}_1(A) \oplus \Sigma_2^\mathrm{p})$.

As an aside, we point out that most of this research was motivated by the problem of determining exactly the power of sets in P/poly as oracles. For example the bound above was a key step in locating P/poly optimally within the Extended Low Hierarchy [19]. Another motivation was provided in [32], by the observation that the problem of computing advice functions is a generalization of that of finding bounded representations.

Theorem 12 *[32] Let $\mathcal{M}$ be a machine class. If some $\mathcal{M}$ machine finds bounded representations for* CIR *or* REPCIR*, then every set A in* P/poly *has advice functions in* $\mathrm{PF}(\mathcal{M}(A))$.

In fact, this theorem holds for other representation classes C if one takes the subclass of P/poly of sets having polynomial-size representations in C (a condition stronger than having polynomial-size circuits). Informally, the theorem means that from a machine finding representations for a uniform class we can build others that find advices for the sets in the nonuniform counterpart. Note that the machine computing advice functions can in principle be different for every set in P/poly. It is therefore surprising that the converse often holds.

Theorem 13 *[32] Let $\mathcal{M}$ be a machine class. If every set A in* P/poly *has advice functions in* $\mathrm{PF}(\mathcal{M}(A))$*, then there are* $\mathrm{PF}(\mathcal{M}())$ *machines that find bounded representations for* CIR *and* REPCIR.

(Taking again subclasses of P/poly, the result holds more in general for classes with a few closure properties. Among these classes are DFA, NFA, and CFG.)

From the upper and lower bound results, we immediately obtain the following corollary.

Corollary 14 *(1) If* CIR *is learnable with any protocol using the queries in Table 1, then every set A in* P/poly *has an advice function in* $\mathrm{PF}(\mathrm{NP}_1(A))$.

(2) REPCIR *is bounded-learnable from Subset and Superset queries if and only if every set A in* P/poly *has an advice function in* $\mathrm{PF}(\mathrm{NP}_1(A))$.

This suggests that resolving learnability of circuits may hinge on improving our bounds on the complexity of advice functions.

Let us observe that the proof of the corollary relativizes to arbitrary oracles. Therefore, together with Theorem 11 it implies that REPCIR is {Sub,Sup}-learnable given an additional Σ_2^{p} oracle.

5 Proper Learning: A Combinatorial Characterization

Most of the learning algorithms designed in the query learning paradigm are proper. On the other hand, for proper learning many negative results, in fact non-identifiability results, are known. As an example of such negative results, we consider the protocol {Mem} and show that most interesting classes are not {Mem}-learnable.

Take any class C that can represent every singleton set $\{x\}$ in size polynomial in $|x|$. This is certainly the case for finite automata, circuits, and formulas. No matter what computing power it may have, any Membership learner for this class needs a number of queries exponential in n in order to identify every concept of size n. Indeed, consider the adversary that keeps answering 'no', and only answers 'yes' after having seen the 2^n different singletons. For a fixed learner, the answers given by this adversary are always consistent with at least one singleton set, and the learner must really ask $2^n - 1$ singletons in order to correctly identify every target concept.

A similar argument can be made for learners using Subset queries besides Membership ones. An abstract version of this argument is given in [32]: no

PF($\mathrm{NP}_{\overline{m}}()$) machine can find bounded representations for classes that can represent singletons with only polynomial blow-up, so, by the upper bounds given in the previous section, these classes are not {Mem,Sub} learnable. The proof is based on a diagonalization exploiting the fact that $\mathrm{NP}_{\overline{m}}()$ machines can only query their oracle negatively. Still a similar argument, now with the complement of singletons, establishes the negative result for Membership and Subset queries; this time one uses $\mathrm{NP}_{m}()$ machines, that can only get positive information from the oracle.

The situation becomes much more interesting for protocols whose simulation seems to need both positive and negative information, such as those containing Equivalence queries or the combination of {Sub,Sup}. There are indeed a few interesting classes known to be learnable from Equivalence queries alone, but Angluin [3] showed that many important classes are not.

Theorem 15 *[3]* DFA*,* NFA*,* CFG*,* DNF*, and* CNF *cannot be identified from a polynomial number of Equivalence queries of polynomial size.*

The proof holds both for exact and bounded learning, but is strongly dependent on the assumption that learning is proper. It is far more complex than for the protocols discussed above, and it goes like follows: first, define a combinatorial property of representation classes, "having approximate fingerprints"; then, show the general result that no class that has approximate fingerprints can be identified from polynomially many polynomial-size queries; finally, show that DFA, NFA, CFG, DNF, and CNF have approximate fingerprints.

Let us state the definition of approximate fingerprints – it is all right if you have to read it several times to get the idea.

Definition 16 *[3] (i) Let T be a class of sets, $w \in \Sigma^\star$ a string, b a value in $\{0,1\}$, and α a real number between 0 and 1. We say that (w,b) is an α-approximate fingerprint for T if*

$$\|\{\ c \in T\ :\ \chi_c(w) = b\ \}\| < \alpha \cdot \|T\|.$$

(ii) Let $C = (R, \Phi)$ be a representation class; a sequence of concept classes is a family $\{T_n\}_{n\geq 0}$, with each T_n a set of concepts in $\Phi(R)$. The sequence is bounded by polynomial p if, for every n, every concept in T_n has size at most $p(n)$.

(iii) We say that class C has approximate fingerprints if there exist polynomials p_1 and p_2 such that for every polynomial q there is a sequence of concept classes $\{T_n\}_{n\geq 0}$ bounded by $p_1(n)$ and infinitely many n such that

- *$T_n^{\leq p_2(n)}$ contains at least two sets, and*
- *for every $r \in R$ of length at most $q(n)$ there is some $w \in \Sigma^{\leq p_2(n)}$ such that $(w, \chi_{\Phi(r)}(w))$ is a $1/q(n)$-approximate fingerprint for $T_n^{\leq p_2(n)}$.*

(There are two minor changes in this definition with respect to Angluin's, needed to obtain the forthcoming theorem. They are not relevant for the proof of Theorem 15).

The intuitive meaning of this definition is the following. An approximate fingerprint for a class of concepts is a pair (w, b) such that most concepts in the class contain w if $b = 1$ and do not contain w if $b = 0$. Thus, knowing that the correct classification of w with respect to the target is b rules out very few concepts in the class as candidates to be the target. What condition (b) says, in essence, is that for every possible Equivalence query there is some w such that most concepts in a certain subclass T_n agree with the target about w. In other words, for every Equivalence query we can make, a wicked teacher can find a counterexample that rules out very few concepts in T_n, thus forcing us to make many queries in order to isolate one of the targets in T_n. The formalization of this adversary argument is the proof of Angluin's general result that a class with approximate fingerprint is not polynomially identifiable.

It turns out that the converse also holds. This definition, apparently devised just for the proof of a non-learnability result, characterizes precisely what can be identified by means of Equivalence queries. Furthermore, for honest classes, it can be shown that identification strategies need not be arbitrarily powerful.

Theorem 17 *[15] The following are equivalent for every honest representation class C:*

1. *C does not have approximate fingerprints.*
2. *C can be identified with polynomially many Equivalence queries of polynomial size.*
3. *C is learnable from Equivalence queries and an additional oracle in* PH.

In fact, (1) and (2) are also equivalent for classes that are not honest.

Besides characterizing what can be {Equ}-identified, this theorem admits another interpretation from the complexity-theoretic viewpoint. Distinguishing identification and learning from Equivalence queries amounts to showing that the PH oracle in (3) is sometimes required. Thus, for classes with approximate fingerprints, an information-theoretic negative result is possible; such are the results in [3]. For a class without approximate fingerprints, any negative result must be conditioned on P $\neq$ NP or an even stronger hypothesis.

Intuition for the proof. That (2) implies (1) is part of the proof of Theorem 15. That (3) implies (2) is trivial. To show that (1) implies (3), assume that C does not satisfy the approximate fingerprint property. The negation of property (b) says that, starting from any class of concepts, there is a possible Equivalence query r for which any counterexample w will discard a non-negligible fraction of candidates, say, a fraction of $1/poly(n)$. A sufficiently powerful strategy can determine this r and query it for Equivalence so that the set of possible targets shrinks fast until only one candidate is left. We can view this strategy as performing a very weak "majority vote" among all remaining concepts. The proof that power in PH suffices uses, among other structural techniques, Stockmeyer's results that #P can be approximated in Δ_3^p [27].

In hindsight, it would not be surprising that similar characterizations exist for any other protocols using the queries in Table 1. It suffices to find the right combinatorial property, essentially saying "a malicious adversary will succeed in forcing many queries on this class", then an adaptation of the proofs in [3, 15] will yield the analog of Theorem 17. For example, Raghavan [25] has identified the property that works for {Mem,Equ}. This is an interesting and potentially useful case, given the extensive use of this protocol and the important questions still open around it.

6 Circuits: A Complexity-Theoretic Barrier

When in the last section we listed some classes that are definitely not learnable from Equivalence queries, we did not include CIR among them. Indeed, these were unconditional, information-theoretic results, and now we know that such a result for circuits needs some assumption (or else, a hard proof!).

Theorem 18 *[15, 11] If* P = NP, CIR *is* {Equ}*-learnable.*

Do not interpret this result as giving any evidence whatsoever that circuits may be {Equ}-learnable. There is in fact strong evidence that this is not the case. Indeed, we have mentioned that every {Equ}-learnable class is also PAC-learnable, and Valiant [28] already observed that circuits are not PAC-learnable assuming pseudo-random functions exist.

The proof in [15] of Theorem 18 exploits in full the assumption that P = NP. With this assumption, we can cast PH computations on inputs of fixed length into polynomial-size circuits. In particular, there are small circuits that even approximate any #P function, again by Stockmeyer's theorem. Using these circuits as Equivalence queries, it is possible to implement an approximate majority voting that reduces exponentially fast the space of possible candidates for the target.

The independent proof of Bshouty, Cleve, Kannan, and Tamon [11] is much more subtle and informative. They show, unconditionally, that circuits implementing approximate majority vote exist. In fact, these circuits abound within an easily described search space, which can be sampled almost uniformly at random with the help of an oracle in NP. This leads to the following, unconditional, results.

Theorem 19 *[11] (1)* CIR *is* {Equ}*-learnable by randomized learners that query an additional* NP *oracle.*

(2) CIR *is* {Equ}*-learnable by deterministic learners with a* Σ_3^{p} *oracle.*

And Theorem 18 follows immediately from (2). Note that by (1) and the observation in [11] that Subset and Superset queries can simulate an NP oracle, we obtain that CIR is learnable from probabilistic {Sub,Sup}-learners.

Another consequence of Theorem 18 is based on some sort of "universality property" of circuits.

Theorem 20 *Assuming* P = NP*, every honest, fixed-length class is learnable from Equivalence queries and hypotheses in* CIR*.*

In fact, with an NP oracle we can force the output of the learner to be in the class, instead of a circuit. The proof is by standard techniques of encoding P computations into polynomial-size circuits, together with Theorem 18. As a consequence, representation-independent negative results are hard to obtain in query learning. We can sometimes prove that a fixed-length class is not properly learnable, maybe even not learnable from another specific class, but proving unconditionally that it is learnable from no other class is as hard as showing P $\neq$ NP.

It is interesting to compare this situation to the PAC-learning case. For PAC, it is known that given an NP oracle for free, every honest class is properly learnable [10]. So in PAC, all the hardness comes from the computational side. In query learning, information-theoretic negative results are sometimes possible for proper learning, but representation-independent negative results have to assume at least P $\neq$ NP.

Of course, the theorem applies only to fixed-length classes because CIR is fixed-length. To extend it to all honest classes, we must take representations for infinite languages, such as a Turing machines. Grouping all of them in a single class forbids honesty, but we can consider a family of classes parametrized by the running-time bound.

Definition 21 *For every polynomial $p(n)$, representation class C_p is defined as (R_p, Φ_p), where*

- $R_p = \{\ M \mid\ M$ *encodes a Turing machine* $\}$.
- $\Phi_p(M) = \{\ x \mid M(x)$ *accepts in time at most* $p(|x|)\ \}$.

Now, each class C_p is honest, and redoing the proofs of Theorems 18 and 20, we obtain the analog results for classes that are not fixed-length.

Theorem 22 *(1) If* P = NP *then every class C_p is* {Equ}*-learnable from some class C_q. (2) Furthermore, if* P = NP*, every honest class C is* {Equ}*-learnable from some C_q.*

7 Exact vs. Bounded Learning

Before concluding, we study in more detail whether the two models of learning that we have used are indeed different. The answer in fact depends on the protocol. The two models coincide for those protocols that allow to detect when learning has succeeded, and differ for all others.

The following theorem was obtained together with David Guijarro.

Theorem 23 *For any honest representation class C and in any protocol that includes (or can simulate) Equivalence queries, C is bounded learnable if and only if C is exactly learnable.*

Proof. In the presence of Equivalence queries, we can assume the following without loss of generality: before a learner gives some r as output, it has asked r as an Equivalence query and received a 'yes' answer.

With this assumption, the "if" part is easy; it suffices to verify that a bounded algorithm that discards its input (n, m) and learns in the exact sense also learns in the bounded sense.

For the "only if" part, let A be a bounded learner for C that runs in time $p(n, m)$. Recall that $A(n, m)$ expects only counterexamples of length at most $q(m)$ and assumes that the target concept has some representation of length n or less; we assume without loss of generality that p and q are known to A.

To learn C exactly, execute $A(i, i)$ in sequence for $i = 1, 2, 3, 4, \ldots$ Stop the simulation of each $A(i, i)$ as soon as one of the following occurs:

1. $A(i, i)$ outputs r: Output r and halt.
2. The teacher returns a counterexample of length $> q(i)$. Try $i + 1$.
3. $A(i, i)$ exceeds time $p(i, i)$, or halts without output. Try $i + 1$.

If this procedure ever halts, clearly it outputs a correct representation for the target, and the running time so far is $\mathcal{O}(\sum_{j=1}^{i} p(j, j)) = \mathcal{O}(i \cdot p(i, i))$. If the simulation of $A(i - 1, i - 1)$ stopped in case 2, then it received a counterexample of length at least $q(i - 1) + 1$. If it stopped in case 3, then $A(i - 1, i - 1)$ must have been given a wrong size bound, so the size of the target concept is at least i. In both cases, the running time is polynomially bounded in the size of the longest counterexample or concept size.

Termination is not so clear; in principle, an unfriendly teacher can stop each $A(i, i)$ by systematically giving a counterexample of length greater than $q(i)$ to the first Equivalence query. But now we can do the following: After $A(i, i)$ stops in cases 2 or 3, we ask as Equivalence queries the at most i representations of size $\log i$. This ensures termination and keeps the running time polynomial.

A more efficient simulation is obtained by trying only the i that are powers of 2. ■

Note that the theorem holds even for so-called *restricted* Equivalence queries – those in which the answer is 'yes' or 'no', with no counterexample returned in the negative case. For protocols that cannot obtain this information from the teacher, bounded and exact learning are different notions. We prove as an example the case of Subset queries. The trick to exploit is that a learner cannot know whether it has obtained the right representation or whether running with a larger size bound would give a different output.

Theorem 24 *There is a honest class that is bounded learnable but not exactly learnable from Subset queries.*

Proof. Define function $s : \mathbb{N} \to \mathbb{N}$ by $s(0) = 0$ and $s(n+1) = 2^{s(n)}$. The claimed representation class C is defined as (R, Φ), where

- $R = \{ 0^{s(i)} \mid i \in \mathbb{N} \}$, and
- $\Phi(0^{s(i)}) = \{ 0^j \mid j \leq s(i) \}$.

Class C is honest and, given an upper bound n on the size of the target concept, one can learn C in time linear in n: just find the maximum i such that $s(i) \leq n$ and $0^{s(i)}$ is in the target; then output $0^{s(i)}$.

On the other hand, assume that there is an exact learner for C that makes only Subset queries. Let $p(n, m)$ be a polynomial bounding its running time. Choose any i such that $s(i+1) = 2^{s(i)} \gg p(s(i), s(i)+1)$, and run the algorithm on target $\Phi(0^{s(i)})$. If the algorithm does not make a Subset query with $0^{s(i+1)}$, it cannot distinguish between targets $\Phi(0^{s(i)})$ and $\Phi(0^{s(i+1)})$ so it fails to learn one of both concepts. If it does query $0^{s(i+1)}$ (or any longer representation) it must take time at least $s(i+1)$, and string $0^{s(i)+1}$ is for sure a valid counterexample. In this case, the time allowed to the algorithm is $p(s(i), s(i)+1)$, which is much less than $s(i+1)$. A contradiction. ■

To prove the analogous result for Membership queries, modify the class so that $\Phi(0^{s(i)}) = \{ 0^{s(j)} \mid j \leq i \}$. Similar and easy changes in Φ separate bounded learning from exact learning for the protocols {Mem,Sub} and {Mem,Sup}.

8 Future Work

Hopefully, by now we have conveyed the idea that structural notions can be useful to clarify the complexity of learning. For the interested reader, here are some questions that might be clarified by ideas similar to the ones presented.

- Characterize the power of protocols {Equ} and {Mem,Equ} by means of upper and lower bounds as in Section 4.
- There is some work done on query learning algorithms that are fault-tolerant, i.e., learn correctly even if the teacher makes a moderate number of errors [4, 5]. Can we justify, again via upper or lower bounds, that fault-tolerant learning is (is not) qualitatively more difficult than error-free learning?
- We have mentioned that Valiant [28] and Kearns and Valiant [17] have shown that, from the hardness of cryptographic functions, one can prove that circuits, dfas, and boolean formulas are not PAC-learnable, so not {Equ}-learnable. Angluin and Kharitonov [6] extended these results to several other classes, even adding Membership queries. However, while circuits give rise to P-complete membership problems, dfas and boolean formulas lie within NC^1. Can we justify that P-complete classes are indeed more difficult to learn than NC^1 ones? More in general, does the "standard" complexity of a class have any effect on its learning complexity?
- A related problem: The current proof that CIR is not learnable from Equivalence queries goes through the probabilistic PAC-model and is

based on cryptographic assumptions that involve probabilistic computation. Since Equivalence learning is deterministic, can we give a direct proof using weaker hypotheses such as $\Delta_2^p \neq \Sigma_2^p$, or perhaps the existence of honest PF functions whose inverse is not in PF?

Acknowledgements

I am grateful to José L. Balcázar, David Guijarro, Vijay Raghavan, and Osamu Watanabe for exciting discussions while working on this subject. I am especially indebted to Osamu for our joint work and for introducing me to learning theory, and to José for comments on a draft of this paper. I also thank Nader Bshouty, Richard Cleve, and Christino Tamon for comments on our common research, and Bill Gasarch and John Shawe-Taylor for comments on the preliminary version.

References

1. D. Angluin: "Learning regular sets from queries and counterexamples". *Information and Computation* **75** (1987), 87–106.

2. D. Angluin: "Queries and concept learning". *Machine Learning* **2** (1988), 319–342.

3. D. Angluin: "Negative results for equivalence queries". *Machine Learning* **5** (1990), 121–150.

4. D. Angluin: "Computational learning theory: survey and selected bibliography". *Proc. 24th ACM Symposium on the Theory of Computing,* 351–369. ACM Press (1992).

5. D. Angluin: "Learning with Queries". *Computational Learning and Cognition. Proc. of the Third NEC Research Symposium* (Princeton, May 1992). SIAM (1993).

6. D. Angluin and M. Kharitonov: "When won't membership queries help?". *Proc. 23rd ACM Symposium on the Theory of Computing,* 444–454. ACM Press (1991).

7. M. Anthony and N. Biggs: *Computational Learning Theory.* Cambridge University Press, 1992.

8. J.L. Balcázar, J. Díaz, and J. Gabarró: *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science, vol. 11. Springer-Verlag 1988.

9. L. Berman and J. Hartmanis: "On isomorphisms and density of NP and other complete sets". *SIAM Journal on Computing* **6** (1977), 305–322.

10. A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth: "Occam's razor". *Information Processing Letters* **24** (1987), 377–380.

11. N.H. Bshouty, R. Cleve, S. Kannan, and C. Tamon: "Oracles and queries that are sufficient for exact learning". *Proc. 7th ACM Conference on Computational Learning,* 130–139. ACM Press (1994).

12. L. Fortnow, S. Jain, W. Gasarch, E. Kinber, M. Kummer, S. Kurtz, M. Pleszkoch, T. Slaman, F. Stephan, and R. Solovay: "Extremes in the degrees of inferrability". *Annals of Pure and Applied Logic* **66** (1994), 21–276.

13. W. Gasarch and C.H. Smith: "Learning via queries". *Journal of the ACM* **39** (1992), 649–675.

14. R. Gavaldà: "Bounding the complexity of advice functions". *Proc. 7th Structure in Complexity Theory Conference,* 249–254. IEEE Computer Society Press (1992).

15. R. Gavaldà: "On the power of equivalence queries". Proceedings of the First European Conference on Computational Learning Theory, Royal Holloway, University of London, december 1993. In: John Shawe-Taylor and Martin Anthony (eds.), *Computational Learning Theory: Euro-COLT'93,* 193–203. The Institute of Mathematics and its Applications Conference Series, new series number 53, Oxford University Press 1994.

16. R.M. Karp and R.J. Lipton: "Some connections between nonuniform and uniform complexity classes". *Proc. 12th ACM Symposium on the Theory of Computing,* 302–309. ACM Press (1980).

17. M. Kearns and L.G. Valiant: "Cryptographic limitations on learning boolean formulae and finite automata", *Journal of the ACM* **41** (1994), 67–95.

18. K. Ko and U. Schöning: "On circuit-size complexity and the low hierarchy in NP". *SIAM Journal on Computing* **14** (1985), 41–51.

19. J. Köbler: "Locating P/poly optimally in the extended low hierarchy". *Proc. 10th Symposium on Theoretical Aspects of Computer Science,* 28–37. Lecture Notes in Computer Science 665. Springer-Verlag, 1993.

20. R.E. Ladner, N.A. Lynch, and A.L. Selman: "A comparison of polynomial time reducibilities". *Theoretical Computer Science* **1** (1975), 103–123.

21. P.D. Laird: "A survey on computational learning theory". *Formal Techniques in Artificial Intelligence: a Sourcebook.* R.B. Banerji (ed.). North-Holland, 1990, 173–215.

22. N. Littlestone: "Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm". *Machine Learning* **2** (1988), 285–318.

23. B.K. Natarajan: *Machine Learning: a Theoretical Approach.* Morgan Kauffman, 1991.

24. L. Pitt: "Inductive inference, DFAs, and computational complexity". *Proc. International Workshop on Analogical and Inductive Inference AII'89*, Lecture Notes in Artificial Intelligence 397. Springer-Verlag, 1989., 18–42.

25. V. Raghavan, personal communication.

26. U. Schöning: *Complexity and Structure.* Lecture Notes in Computer Science 211. Springer-Verlag, 1986.

27. L. Stockmeyer: "On approximation algorithms for #P". *SIAM Journal on Computing* **14** (1985), 849–861.

28. L.G. Valiant: "A theory of the learnable". *Communications of the ACM* **27** (1984), 1134–1142.

29. M.K. Warmuth: "Towards representation independence in PAC learning". *Proceedings of the International Workshop on Analogical and Inductive Inference AII-89,* Lecture Notes in Artificial Intelligence 397. Springer-Verlag, 1989., 78–103.

30. O. Watanabe: "A formal study of learning via queries". *Proc. 17th International Colloquium on Automata, Languages, and Programming,* 139–152. Lecture Notes in Computer Science 443. Springer-Verlag, 1990.

31. O. Watanabe: "A framework for polynomial time query learnability". *Mathematical Systems Theory* **27** (1994), 211–229.

32. O. Watanabe and R. Gavaldà: "Structural analysis of polynomial time query learnability". *Mathematical Systems Theory* **27** (1994), 231–256.