Some Results in Exact Learning of Boolean Functions

Jorge Castro* Ricard Gavaldà† David Guijarro†

june 2001

Abstract

We describe some recent or not-so-recent results in the model of learning known as exact learning from queries. Some results are chosen because they led to a result in complexity theory, and others are chosen because they seem to indicate an interesting line of research at the common ground of learning and complexity. More precisely, we survey: (1) an improvement of the Karp-Lipton collapse that stemmed from a learning algorithm for Boolean circuits, (2) recent advances in distinguishing information-theoretic from complexity-theoretic difficulty of learning, and (3) recent results on learning small-depth circuits with modular gates that call for a systematic study of the expressive power of so-called Multiplicity Automata.

1 Introduction

Computational learning theory is concerned with the study of formal models of learning with an emphasis in the design of algorithms that provably satisfy some learning criteria and make efficient use of computational resources. From its earliest days, about fifteen years ago [Val84, Ang87], it has drawn problems, concepts, and tools from several different areas (statistics, inductive inference, pattern recognition, machine learning, algorithmics...).

Complexity theory has been central to this field from its birth. Even in the first definitions of the field, a clear premise is that the study of what is efficiently learnable has to be solidly planted in complexity-theoretic basis.

And, indeed, computational learning theory has drawn many complexity-theoretic concepts. NP-completeness and non-approximability results have been used often to show that some representation-dependent learning tasks are infeasible [PW90]. Cryptographic assumptions are at the heart of all representation-independent negative results known so far for PAC learning and its variants [AK95]. An important algorithm for PAC-learning AC^0 under the uniform distribution [LMN93] heavily relies on previous lower-bound results for AC^0 circuits. Examples are too numerous to be all cited here.

In this column we want to review two or three learning results (some quite recent, others a few years old) that have worked or might work the other way: they inspired a complexity-theoretic result, or they may suggest an alternative way of looking at specific complexity issues, or, at least and hopefully, have enough complexity-theoretic flavor to interest some readers of this column.

*Department of Software (LSI), Universitat Politècnica de Catalunya. Jordi Girona Salgado 1–3, E-08034 Barcelona, Spain. {castro|gavalda}@lsi.upc.es. Partially supported by the EU ESPRIT Working group NeuroCOLT2, the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT), and by FRESCO PB98-0937-C04-04.

†Mannes Technology Consulting, Pl. Tirant lo Blanc 7, E-08005 Barcelona, Spain. david@mannes-tech.com.

Certainly, there has been more feedback from learning theory into complexity than the one described here; we do not intend to be exhaustive by any means. In particular, the results we present deal with the Angluin's model of exact learning from queries, rather than Valiant's PAC model that often deserves more attention in expositions of learning theory.

We start by describing a result of Watanabe, based on a learning algorithm by Bshouty *et al.*, that improved the known collapse of the polynomial-time hierarchy, under the hypothesis that NP has polynomial-size circuits, due to Karp and Lipton. The reason for choosing to describe this result is two-fold. First, because the authors are periodically asked to sketch its proof by other complexity theorists. Second, because it provides a nice introduction to the next topic: The investigation of how impossibility results in learning may be due to two different causes, information hardness or computation hardness. We describe the origins of such investigation and some recent advances that, in some sense, have essentially closed the issue. Finally, we change the topic a bit and discuss the status of learning constant-depth circuits using modular gates. We describe the results obtained so far, how they compare to size lower bound results, discuss the concept of multiplicity automata, which has been the key for these learning results, and advocate a complexity-theoretic study of multiplicity automata as a possible way of improving our understanding of some of these circuit classes.

Because of space limitations, many definitions and technical points are omitted. We hope however that the pointers to the literature are useful to the interested reader.

Acknowledgements. We are grateful to Lane Hemaspaandra for inviting us to write this survey, and for telling us about Cai's recent work [Cai01]. We also thank Jin-Yi Cai for promptly sending us a copy of [Cai01].

2 Preliminaries

We review the concepts of learning theory that we need strictly for the purpose of this article. In particular, we discuss mostly the model of exact learning from queries introduced by Angluin [Ang87, Ang88], and furthermore we specialize it to the case of learning Boolean functions. That is, all the concepts we consider are functions $\{0,1\}^n \to \{0,1\}$ for some n. The reader should be aware that the general model accounts for language learning (i.e., a concept may contain strings of any length), as well as learning over other domains.

Precise definitions and background can be found in the surveys and textbooks listed in the COLT repository (`www.learningtheory.org`). The textbook whose presentation is closest to ours is the one by Kearns and Vazirani [KV94].

Exact learning is a game between two players, the learner and the teacher. The teacher has a Boolean function $f : \{0,1\}^n \to \{0,1\}$ in mind, that we call the *target function*. During the game, the learner poses *queries* about f to the teacher, and the teacher returns *answers*. The teacher is faithful, in the sense that answers are correct with respect to f, but adversarial, in the sense that, whenever more than one answer is correct, it can choose to return that one that is least helpful to the learner. The goal of the learner is to identify exactly the function f, and to do so as quickly as possible.

We usually think of the learner as an algorithm (hence, we care about its use of resources such as time and memory), while the teacher is just an strategy for answering queries (hence, we do not care about how hard it is to implement it).

Learner and teacher agree beforehand to a notion of "function complexity" s. Efficiency of the learner is measured as a function of the number of variables, n, and of the complexity of the target function f, call it $s(f)$. Reasonably enough, the learner should be allowed more time to win the game for larger values of n and $s(f)$. In general, we want to design algorithms whose running time is bounded by a polynomial of n and $s(f)$ for any function f and any teacher for f, no matter how adversarial.

Measure $s(f)$ is usually formalized by fixing a representation method for Boolean functions, such as DNF formulas, acyclic DFA, or Boolean circuits. Then the complexity of f is the size of the smallest representation for f in that method. This representation method is called the "target class" of the algorithm, call it F during this discussion. The name "class" arises because we can also consider representation methods that are not universal, that is, that cannot represent all Boolean functions. In this case, fixing that representation method indeed limits the class of allowed target functions and provides additional information to the learner. For example, if our target class is that of of monotone Boolean circuits, then the learner can safely assume that the target function f is monotone, and its complexity $s(f)$ is by definition the size of a smallest monotone circuit for f.

We said before that the goal of the teacher is to identify f exactly, but did not say what "identifying" means. Precisely, the learner is required to output an encoding of function f in some (other) representation method for boolean functions, called the "hypothesis class" and denoted here by H. Target class and hypothesis class may or may not be the same. Still, if we want to allow learning at all, H should contain representations for all functions in F; for example, it makes little sense to choose F to be general circuits and H to be monotone circuits. And, if we want to allow for polynomial-time learning, each function in F must have some representation in H that is at most polynomially larger. Additionally, we want to consider only classes H and F that can be evaluated in polynomial time, so they should be no more expressive than Boolean circuits.

To define the learning game completely, learner and teacher have to agree on the types of queries are allowed to the learner and the valid answers from the teacher. This is called the *protocol* of the learning game. We consider two standard query types in Angluin's model, Membership and Equivalence queries. Let the target function be f. In a Membership query, the learner produces an element $x \in \{0,1\}^n$ and the teacher must return $f(x)$. In an Equivalence query, the learning algorithm produces a hypothesis $h \in H$ in the hypothesis class; the teacher must return 'Yes' if $h \equiv f$, or else a counterexample: any $x \in \{0,1\}^n$ such that $f(x) \neq h(x)$. Occasionally we mention Subset queries (where the query is a hypothesis h and the answer is 'Yes' iff h logically implies f, or else a counterexample) and the dual Superset queries [Ang88].

All in all, a learning result has to state the target class being learned, the hypothesis class involved, and the protocol used. As an example, Angluin [Ang88] proved that monotone DNF formulas (target class) are learnable from Membership and Equivalence queries (protocol) with monotone DNF formulas (hypothesis class).

3 Learning circuits exactly improves the Karp-Lipton collapse

The distinction between nonuniform and uniform computation models has been present in complexity theory from its origins. It was also clear from the early days that, given the same resources, nonuniform models can do at least as much as uniform ones, and often more.

In a classical paper, Karp and Lipton [KL82] first showed a connection in the opposite direction: often, assumptions about the nonuniform complexity of problems have consequences concerning uniform complexity. A prime example of this connection is the following theorem.

Theorem 1 *[KL82] If* NP *has polynomial-size circuits, then* $\mathrm{PH} = \Sigma_2^p$.

For brevity, we refer to this kind of results as "the Karp-Lipton collapse". It is known that there are oracles relative to which NP has polynomial-size circuits yet $\mathrm{PH} \neq \Delta_2^p = \mathrm{P}^{\mathrm{NP}}$, so the Karp-Lipton collapse cannot be improved to Δ_2^p by techniques that relativize. Thus, for twelve years the collapse remained at the Σ_2^p level.

In 1994, with a different goal in mind, Bshouty *et al.* [BCKT94] presented an exact learning algorithm for Boolean circuits. Right after that, Osamu Watanabe observed that this learning algorithm gives the following improvement of the Karp-Lipton collapse:

Theorem 2 *(Watanabe, in [BCG$^+$96]) If NP has polynomial-size circuits, then* $\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}}$.

Later, Köbler and Watanabe extended the result and reformulated it in terms of self-reducibility and lowness [KW99], and without learning terminology. Next we review the original proof of this result with its learning ingredient.

Here is the proof of Karp-Lipton's result in a nutshell: Let L be a Σ_i^p language. Membership to L for strings of length n can be decided by a Σ_{i-1}^p predicate if one has a circuit that correctly decides $SAT^{p(n)}$ in one's hand (for some polynomial p). Hence, if circuits for SAT can be produced within Σ_{i-1}^p resources, L is in Σ_{i-1}^p.

In [KL82], circuits for $SAT^{p(n)}$ are just guessed and verified: The direct way of checking that a circuit C accepts $SAT^{p(n)}$ (e.g., for all F of length $p(n)$, F is in SAT iff C accepts F) is a Π_2^p predicate, hence circuits can be produced in Σ_3^p and we can conclude that $\Sigma_4^p = \Sigma_3^p$. Karp and Lipton observed that, using the downward self-reducibility of SAT, one can turn this check into a coNP predicate, and from here the result that $\Sigma_3^p = \Sigma_2^p$ follows.

Of course, more efficient ways (than guess-and-verify) of producing circuits for SAT would improve the result. After seeing the result in [BCKT94], Watanabe observed that learning the circuits provides an alternative way. Precisely, he observed:

Fact. If SAT has polynomial-size circuits, and if Boolean circuits are learnable from Equivalence queries by a probabilistic, polynomial-time algorithm with additional access to an NP oracle, then circuits for SAT can be computed in $\mathrm{ZPP}^{\mathrm{NP}}$.

Note that the learning algorithm in the premise has two oracles – the Equivalence query oracle for an unknown Boolean function *and* an NP oracle –, while the circuit producer in the conclusion has only one oracle, i.e., SAT.

Also, compare this fact with the observation in [IW98] that functions that are both self-reducible and learnable from Membership queries are in BPP.

Why is this fact true? Again, self-reducibility of SAT is the key ingredient. Suppose inductively that we have a circuit of size $p(m)$ that correctly solves instances of SAT of length m. Then, using self-reducibility, we can also build a circuit for instances of length $m+1$ of size about $2p(m)$. To avoid an exponential growth, use the learning algorithm to obtain a new circuit of size $p(m+1)$, answering Equivalence queries by means of the $2p(m)$-size circuit. Solving Equivalence queries with circuits is easily achieved in polynomial time with an NP oracle.

Finally, let us sketch the learning algorithm for Boolean circuits. Fix a learning algorithm using Equivalence queries and a teacher for an unknown target function f. Define S_t as the set of pairs $(x, f(x))$ received as counterexamples after t queries, and $CONS(S_t, m)$ be the set of circuits of size m consistent with S_t. That is, a circuit C of size m is in $CONS(S_t, m)$ if $C(x) = f(x)$ for all pairs $(x, f(x))$ in S_t. The goal of the learning algorithm is to ask at time t a query such that $CONS(S_t, m)$ is substantially smaller than $CONS(S_{t-1}, m)$, say c times smaller for a constant $c < 1$, no matter what an adversarial counterexample is returned by the teacher. If the algorithm manages to do this for $m \log(1/c)$ times, it has ensured that only circuits computing exactly f remain.

Call a point $x \in \{0,1\}^n$ *undisputed* at round t if there is a clear bias among circuits in $CONS(S_{t-1}, m)$ towards accepting or rejecting x, e.g., either a fraction of at least 3/4 of circuits $CONS(S_{t-1}, m)$ accept x, or a fraction of 3/4 of these circuits reject x. Conversely, point x is called *disputed* if the fraction of circuits in $CONS(S_{t-1}, m)$ accepting x is between 1/4 and 3/4.

Disputed points make very good counterexamples: Whenever the teacher returns a disputed point as a counterexample for the tth query, $CONS(S_t, m)$ has size at most 3/4 times that of $CONS(S_{t-1}, m)$. The key idea of the learning algorithm is how systematically produce queries that also make progress when they receive undisputed points as counterexamples. The following probabilistic procedure yields a circuit C that has this property with high probability: Draw k circuits $C_1, \ldots, C_k$ from $CONS(S_{t-1}, m)$, independently and uniformly at random; let $C(x)$ be the circuit that takes the majority $C_1(x), \ldots, C_k(x)$. This is a circuit of size $km + O(k \log k)$. It is easy to prove using Chernoff bounds that, for $k = O(n)$ and with very high probability, C classifies all undisputed points as the majority of circuits in $CONS(S_{t-1}, m)$ does. Therefore, if C is asked as an Equivalence query and an undisputed counterexample is received, that counterexample proves wrong a majority (in fact, a fraction of at least 3/4) of the circuits in $CONS(S_{t-1}, m)$, and the learner makes progress as required.

To sample circuits from $CONS(S_{t-1}, m)$, the learning algorithm can use the procedure in [JVV86]: this is a probabilistic, polynomial-time procedure using an NP oracle, and provides a sufficiently close approximation to the uniform distribution for our purposes. Note that this sampling phase in itself accounts for the complexity of the order of $\mathrm{ZPP}^{\mathrm{NP}}$.

Back to learning, what is the significance of this learning algorithm, and why did Bshouty *et al.* cared to develop it? Its requirement for an NP oracle seems to make it essentially inapplicable ... as is to be expected, given that Boolean circuits are provably not learnable under cryptographic hypothesis [AK95]. This is true not only for Equivalence query learning but in all reasonable models of polynomial-time learning. The purpose of exhibiting this and similar algorithms is to tell apart two possible causes of impossibility results in query learning: the need for a lot of interaction between teacher and learner, and the need for the learner to perform expensive computations internally. We review in the next section what is known in this direction.

Note: After the first version of this note was submitted, the editor pointed to us that very recently an even stronger form of the Karp-Lipton collapse has been obtained. Samik Sengupta's observed [Cai01] that Karp and Lipton's proof in fact provides a collapse to the symmetric alternation class S_2 [Can96, RS98]. Jin-Yi Cai has proved [Cai01] that S_2 is included in $\mathrm{ZPP}^{\mathrm{NP}}$, and therefore obtained a collapse to a class presumably smaller than $\mathrm{ZPP}^{\mathrm{NP}}$.

4 Query learning and learning complexity

The issue we just mentioned is void for PAC learning of Boolean functions. It is known that, with the help of an oracle in NP, all reasonable classes of Boolean functions are PAC learnable from a polynomial number of examples. This is because PAC learning is no harder than the computational problem of finding a hypothesis in the target class that is consistent with a polynomial number of examples, and this is always possible in polynomial time with an NP oracle. So in PAC learning, all negative results must assume at least $\mathrm{P} \neq \mathrm{NP}$.

(Note: this discussion applies only to *Boolean* function learning only. In other kinds of function learning, such as those involving real numbers, there is an issue of information vs. computation hardness in PAC learning. There, VC-dimension [KV94] measures the amount information that a PAC learner must obtain in order to learn. In general, and very roughly speaking, a class is PAC learnable from a polynomial number of examples iff its VC-dimension is polynomially growing, and it is a fact that all classes of Boolean functions have polynomially growing VC-dimension.)

The picture, as we will see, is rather different in the exact learning case: there are cases where not only the time needed to learn but also the amount of interaction may have to be superpolynomial.

In the setting of query learning, it is thus interesting to distinguish between the number of queries or game rounds (query complexity) needed to complete the game and the computation time required on the learner's side (time complexity). Since the former is a lower bound of the latter, it has been considered as a separate complexity measure. Ideally, when studying the learnability of some concept class in a given protocol, one analyzes whether the query complexity is low enough (say, polynomial). If so, it makes sense to dedicate extra effort to finding a time-efficient algorithm. Otherwise, the only hope is to change the problem, either by considering a different protocol, by restricting the target class, or by enlarging the hypothesis class.

The distinction between information-cost and computation-cost should not be new to complexity theorists. Communication complexity studies the amount of interaction needed to compute a function whose input is shared among several parties, disregarding the computation costs internal to each party. Decision-tree complexity measures the number of elementary tests to be made on the input in order to decide something about it.

This distinction was made first for learning from Equivalence queries only. A combinatorial notion called "approximate fingerprints" was first shown to be a lower bound [Ang90] and later [Gav94] an upper bound on the number of Equivalence queries required to learn a class. Another notion called "certificates" [HPRW96] was shown to play the same role with Membership and Equivalence queries. These combinatorial notions were designed to distinguish polynomial from superpolynomial learnability only. Later, The "extended teaching dimension" [Heg95] was defined as a function and gives an approximation of the number of Membership queries needed and sufficient to learn.

A unifying step was taken in [BCG00], in order to capture the essence of all the dimensions discussed so far. There, a combinatorial quantity called the *Abstract Identification Dimension* (or AIdim) was presented, and shown to provide a reasonably good approximation of the query complexity in a wide variety of protocols. In particular, it covers all *example-based* queries, those whose answers can all be described by a set of labeled examples, pairs of the form $(x, f(x))$. Clearly protocols comprising any combination of Equivalence, Membership, Subset, and Superset queries are all example-based. In this situation, a majority-based algorithm similar to the one described in Section 3 shows that query complexity is not much larger than AIdim.

Although a large part of the literature on query learning uses example-based queries, other queries have been introduced with specific purposes, such as modeling partial observability of the target, or queries that provide no counterexample. For these, AIdim fails to capture the exact query complexity of learning problems. The result that we are going to present appears in [BCG01], and is the natural continuation and completion of the work described before: it yields a function that, for any target concept class and any (reasonable) protocol, returns a reasonably good approximation of the query complexity. Interestingly, at this level of abstraction majority-based algorithms may not necessarily be optimal learners.

The general idea is the following: consider, associated to every answer, the set of concepts that may be consistent with it (the meaning of "consistent" may depend on the protocol). There is of course a trivial way to express the number of queries required to learn the class. Consider a statement that describes learning as a game where the "learner" player tries to isolate a single target concept: there exists a query such that for any answer to it there is another query such that for any answer to this one, etc… such that at most one concept in the target class is consistent with all the answers. Let d be the minimum number of quantifier alternations such that this statement is true. It should be no surprise that d is the query complexity of the target class with respect to the protocol. Is there a more manageable expression for it? The result that we will present "flattens" this expression to simply: there are d answers that are consistent with at most one target concept.

To do this flattening, however, we need to consider non-adaptive strategies for answering queries. An *answering scheme* is any way of choosing at least one answer for every possible query. Observe only that an answering scheme need not be consistent with any function in the target class, while a teacher for a target function is by definition consistent with that function.

Now let C be a target class of Boolean functions, and P a protocol. Let $d \geq 0$ be the minimum positive integer such that for any answering scheme there always exists at most d answers whose intersection with the target class is at most one concept. If this number exists we say that $\mathrm{Gdim}(C, P)$ is d, otherwise $\mathrm{Gdim}(C, P)$ is ∞.

The following theorem states that Gdim is a right notion of learning dimension. For a class C and protocol P, let $\mathrm{QC}(C, P)$ denote the query complexity of learning C under protocol C, that is, the maximum number of queries used by an optimal learning algorithm with unbounded computation power to learn functions in C under protocol P; in general, this is to be taken as a function of the arity of target functions and their representation sizes. The reader can find formal definitions and a full proof in [BCG01].

Theorem 3 *For any target class C and any protocol P,*

$$\mathrm{Gdim}(C, P) \leq \mathrm{QC}(C, P) \leq \lceil \ln |C| \rceil \cdot \mathrm{Gdim}(C, P).$$

Thus, learning is a somewhat atypical game: The development of d rounds of the game can be described not only as a chain of d alternating quantifiers, but also by a Σ_2 - style sentence. This situation may have a topological interpretation; a first step towards an explanation of query complexity in terms of topological compactness can be found in [Bal99].

5 On circuits with modular gates

Understanding the power and limitations of different computation models is the essence of complexity theory. Since this is difficult, one usually considers a range of computation models of increasing complexity and gradually tries to understand more and more complex models in the range. Small-depth circuits are particularly apt for this kind of program, as one can obtain a very fine range of models by varying not only depth but also circuit topology and gate types. "Understanding the limitations" a class of small-depth circuits, is intended to mean, in most cases, proving some size lower bound, or some size-depth tradeoff.

With the same idea of making gradual progress, learning theory has also tried to understand larger and larger families of constant-depth circuits. In learning theory, "understanding" means ..., well, learning, although one can consider different learning protocols and models to add a little variety to the question. Alternatively, for some families of circuits one can try to prove, unconditionally or conditionally, that they are not learnable.

Have complexity theory and learning theory followed similar paths in the study of small-depth circuits? In this section we want to highlight the differences and similarities, and encourage a line of study that might be helpful for both fields.

Let us consider first AC^0-like circuits. Strong and tight lower bounds are known for circuits containing only negations and unbounded fan-in AND and OR gates, mainly using the random restriction method. Although some questions remain (e.g., finding alternative proof techniques), these circuits are essentially understood.

With respect to learning, the question is far from solved. A remarkable result is the learning algorithm by Linial, Nisan, and Mansour [LMN93] that learns AC^0 circuits in quasipolynomial-time in a model similar to PAC. Note that the result strongly uses the lower bounds for AC^0 circuits, in addition to several other clever ideas. Unfortunately, the PAC learning model in this result is not standard, as examples are assumed to arrive from the uniform distribution, and this is often considered too strong an assumption for learning. So learning AC^0 circuits has to be considered wide open. In fact, even learning depth-two AC^0 circuits (e.g., DNF or CNF formulas) in a standard model of learning has resisted intense effort and, although several learnable subclasses have been identified, some truly new idea seems to be required as of now. Again, Jackson [Jac97] gave a quite sophisticated learning algorithm for DNF which is PAC-style and uses Membership queries, but also requires the assumption that examples are chosen from the uniform distribution.

So complexity theory proceeded next to constant-depth circuits with more powerful gates, such as threshold gates and modular gates. In the case of threshold gates only, lower bounds for really small depths are known [HMP+87]. In learning, basically only depth-one threshold circuits (perceptrons) are known how to be learnable; on the other hand, it can be shown that from some fixed depth on, threshold circuits are definitely not learnable under the assumption that specific cryptographic tasks are hard [AK95].

Let us consider now circuits made of modular gates only; we omit in this discussion ACC^0-type circuits, where both modular gates and AND and OR gates are allowed. To discuss the results here, one has to be quite careful on how modular gates are defined. So, for natural numbers n and m and for a subset $A \subseteq \{0, \ldots, m\}$ define the function (and gate) MOD_m^A to be:

$$MOD_m^A(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } (\sum_{i=1}^n x_i \bmod m) \text{ is in } A \\ 0 & \text{otherwise} \end{cases}$$

The following lower bounds are known:

LB1. Krause and Waack [KW91] proved that depth-2 circuits built as one $MOD_m^{\{1,2,\ldots,m-1\}}$ gate connected to any number of symmetric gates require exponential size to compute the sequence identity problem, for any m (even non-constant). Note that all MOD_m^A gates defined above are symmetric. The result is proved using communication complexity and an argument about the rank of communication matrices of this type of circuits.

LB2. Barrington, Straubing, and Thérien [BST90] proved that circuits made as one MOD_p^A gate connected to MOD_m^B gates needs exponential size to compute the AND_n function, for any A, B, and m but for p prime only. The proof uses algebraic techniques.

LB3. Krause and Pudlák [KP93] prove that circuits built as a $MOD_{p^k}^{\{0\}}$ gate connected to any number of $MOD_m^{\{0\}}$ gates require exponential size to compute AND_n, for prime p and any m. The proof uses several techniques, including random restrictions, and voting polynomials.

LB4. Quite recently, Grolmusz and Tardos [GT00] generalized LB2 and LB3 in the following way: Circuits consisting of any constant number of levels of $MOD_{p^k}^A$ gates, and one layer of MOD_m^B gates connected to the inputs requires exponential size to compute the AND_n function, for any A, B, and m, and prime p.

For learning, the following is known for the model of polynomial-time learning with Membership and Equivalence queries:

LT1. Bergadano, Bshouty, Tamon, and Varricchio [BBTV97] show that depth-2 circuits made of a $MOD_p^{\{1,2,\ldots,m-1\}}$ gate (prime p) connected to any number of $MOD_{m_i}^{\{1,2,\ldots,m-1\}}$ gates (with any combination of different moduli m_i in the circuit) are learnable. These circuits are a special case of those in LB1.

LT2. Bshouty , Tamon, and Wilson [BTW98] show that constant depth Boolean circuits made of $MOD_{p^k}^{\{0\}}$ gates are learnable. Note that these circuits are a subclass of those in LB4.

Both results LT1 and LT2 rely on the same learning technique: a Membership and Equivalence learning algorithm due to [BV96] to learn so-called *Multiplicity Automata*. In fact, what is particular to results LT1–LT2 is the proof that functions computed by these classes of circuits are also computed by polynomially larger Multiplicity Automata.

Multiplicity Automata (MA for short) are a natural generalization of classical automata, well studied from the automata-theoretic point of view. Several equivalent definitions are possible; we give only one and refer the reader to [BV96, BBB+00] for a more detailed treatment. Let K be a ring. An MA over ring K and alphabet Σ is a nondeterministic finite automata M over Σ where each transition is additionally labeled by an element in K. To each transition path in the automata we associate a value, the product (in K) of the elements along its edges. The multiplicity automata defines a function $f : \Sigma^\star \to K$ as follows: On input x, consider the set of all nondeterministic paths in M induced by x; $f(x)$ is simply the sum of the values associated to these paths.

Bergadano and Varricchio [BV96] gave a polynomial-time learning algorithm to learn MA over *fields* from Membership and Equivalence queries (see [BBB+00] for a more efficient version). This algorithm suddenly opened the door to learning a variety of new classes, some of which had resisted substantial efforts before. Most notably, it implies learnability of unambiguous NFA [BV96], sat-k DNF formulae, polynomials over finite fields, bounded-degree polynomials over infinite fields, exclusive-OR of terms, and special cases of decision trees [BBB+00]. Papers [BBTV97] and [BTW98] contain, besides results LT1 and LT2, several others relying on the MA algorithm. In all cases, the expressive power of MA is the key ingredient, in the sense that the learning algorithm remains untouched and the the task is to show that MA have at least as much as expressive power as the class at hand.

One possible line of extension of LT1–LT2, which might bring them closer to LB1–LB4, is to make them work for non-prime moduli. For moduli that are not even prime powers, this should involve dealing with rings instead of fields. Bshouty, Tamon, and Wilson [BTW98] extended the field-MA learning algorithm to work over a large class of rings, including all finite rings. This new algorithm has not provided, so far, as many new learning results, but it contains some truly new ideas on top of the field case that it may prove useful in the future.

Let us sketch one particular application of the ring algorithm. Gavaldà and Thérien have studied the problem of learning expressions and programs over monoids [GT01]. For a monoid G, an expression E over variables $x_1, \ldots, x_n$ is just a string over $\{x_1, \ldots, x_n\} \cup G$. The expression computes a function $G^n \to G$ in the obvious way: on an assignment $a_1 \ldots a_n$ to variables, replace each occurrence of x_i with a_i, and multiply in G the resulting string of G-elements. A program is a generalization of this model where each position in the expression can apply a different mapping $G \to G$ to the variable it reads before passing it for multiplication.

In [GT01], the complexity of learning expressions and programs is shown in many cases to be related to the algebraic complexity of monoid G. As an example, of G is a group of the form $Z_a \circ Z_b$ (read: an extension of Z_b by Z_a, a special case of wreath product of cyclic groups), then expressions over G can be described by a pair formed by a linear expression over Z_b and a MA over the ring Z_a, for any a and b. In particular, the algorithm in [BTW98] for learning MA over rings can be used to learn expressions over $Z_a \circ Z_b$ groups, and in fact the result can be made to work for programs in addition to expressions. Interestingly, when used to compute Boolean functions [BST90, BT88], programs over $Z_a \circ Z_b$ groups are as powerful as circuits consisting of a MOD_a^A gate over MOD_b^B gates, for any a, A, b, and B, for which we are aware of no lower bounds. Unfortunately, we do not know whether learning programs over G as studied in [GT01] implies learning the family of Boolean circuits associated to G. But this result suggests that these families of circuits may admit analysis and that MA may have a role to play there.

This role may be there even for some existing lower bound arguments. We are not in a position to make any formal conjecture, but just observe the following: after all, many of these arguments involve either properties of polynomials or ranks of communication matrices. MA are easily seen to be generalizations of polynomials: a polynomial with t terms is equivalent to an MA with t disjoint branches starting at different initial states. As for matrix rank, it is known that for any function $f : \Sigma \to K$, the number of states in the minimal MA computing f is precisely the rank of the *Hankel matrix* of f, which, in some sense, describes f as completely as its communication matrix – see [BBB+00] for definitions.

All in all, we believe that a careful study of the expressive power of Multiplicity Automata can lead to interesting conclusions; if not to new results, at least to a better understanding of the connections between learning and complexity of small-depth modular circuits.

References

[AK95] D. Angluin and M. Kharitonov. When won't membership queries help? *Journal of Computer and System Sciences*, 50:336–355, 1995.

[Ang87] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[Ang90] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.

[Bal99] J.L. Balcázar. The consistency dimension, compactness, and query learning. In *Computer Science Logic'99*, pages 2–13. Springer-Verlag Lecture Notes in Computer Science 1683, 1999.

[BBB+00] A. Beimel, F. Bergadano, N.H. Bshouty, E. Kushilevitz, and S. Varricchio. Learning functions represented as multiplicity automata. *Journal of the ACM*, 47:506–530, 2000.

[BBTV97] F. Bergadano, N.H. Bshouty, C. Tamon, and S. Varricchio. On learning branching programs and small depth circuits. *Proc. 3rd European Conference on Computational Learning Theory (EuroCOLT'97), Springer-Verlag LNCS*, 1208:150–161, 1997.

[BCG+96] N.H. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52:421–433, 1996.

[BCG00] J.L. Balcázar, J. Castro, and D. Guijarro. A new abstract combinatorial dimension of exact learning via queries. In *Proc. 13th Annu. Conference on Comput. Learning Theory*, pages 248–254. Morgan Kaufmann, San Francisco, 2000.

[BCG01] J.L. Balcázar, J. Castro, and D. Guijarro. A general dimension for exact learning. Technical report, ALCOMFT-TR-01-21, available from `http://www.brics.dk/cgi-alcomft/db`, 2001.

[BCKT94] N. H. Bshouty, R. Cleve, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. In *Proc. 7th Annu. ACM Conf. on Comput. Learning Theory*, pages 130–139. ACM Press, New York, NY, 1994.

[BST90] D.A. Mix Barrington, H. Straubing, and D. Thérien. Non-uniform automata over groups. *Information and Computation*, 89:109–132, 1990.

[BT88] D.A. Mix Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35:941–952, 1988.

[BTW98] N.H. Bshouty, C. Tamon, and D.K. Wilson. Learning matrix functions over rings. *Algorithmica*, 22:91–111, 1998.

[BV96] F. Bergadano and S. Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. *SIAM J. Comput.*, 25(6):1268–1280, 1996.

[Cai01] Jin-Yi Cai. $S_2^p \subseteq \mathrm{ZPP}^{\mathrm{NP}}$. Manuscript, 2001.

[Can96] R. Canetti. On BPP and the polynomial-time hierarchy. *Information Processing Letters*, 57:237–241, 1996.

[Gav94] R. Gavaldà. On the power of equivalence queries. In *Computational Learning Theory: EuroColt '93*, volume New Series Number 53 of *The Institute of Mathematics and its Applications Conference Series*, pages 193–203, Oxford, 1994. Oxford University Press.

[GT00] V. Grolmusz and G. Tardos. Lower bounds for $(\mathrm{MOD}_p$–$\mathrm{MOD}_m)$ circuits. *SIAM Journal on Computing*, 29:1209–1222, 2000.

[GT01] R. Gavaldà and D. Thérien. Learning expressions over monoids. In *Proc. 18th Intl. Symposium on Theoretical Aspects of Computer Science (STACS'01)*, pages 283–293. Springer Verlag Lecture Notes in Computer Science 2010, 2001.

[Heg95] T. Hegedüs. Generalized teaching dimensions and the query complexity of learning. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 108–117. ACM Press, New York, NY, 1995.

[HMP+87] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, and G. Turán. Threshold circuits of bounded depth. In *Proc. 28th IEEE Symp. on Foundations of Computer Science*, pages 99–110. IEEE Computer Society Press, 1987.

[HPRW96] L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan, and D. Wilkins. How many queries are needed to learn? *Journal of the ACM*, 43:840–862, 1996.

[IW98] R. Impagliazzo and A. Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Proc. 39th Annu. Symp. on Foundations of Computer Science*, pages 734–743. IEEE Computer Society Press, 1998.

[Jac97] J.C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *J. Comput. Syst. Sci.*, 55(3):414–440, 1997. Earlier version appeared in FOCS '94.

[JVV86] M.R. Jerrum, L.G. Valiant, and V.V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[KL82] R.M. Karp and R.J. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28:191–209, 1982.

[KP93] M. Krause and P. Pudlák. On the computational power of depth 2 circuits with threshold and modulo gates. In *Proc. 25th ACM Symp. on the Theory of Computing*, pages 48–57. ACM Press, 1993.

[KV94] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, Cambridge, Massachusetts, 1994.

[KW91] M. Krause and S. Waack. Variation ranks of communication matrices and lower bounds for depth two circuits having symmmetric gates with unbounded fan-in. In *Prpc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 777–787. IEEE Computer Society Press, 1991.

[KW99] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. *SIAM Journal on Computing*, 28:311–324, 1999.

[LMN93] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM*, 40:607–620, 1993.

[PW90] L. Pitt and M. K. Warmuth. Prediction preserving reducibility. *J. of Comput. Syst. Sci.*, 41(3):430–467, December 1990.

[RS98] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Computational Complexity*, 7:152–162, 1998.

[Val84] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.