On Computational Hardness in Query Learning

Satwik Bhattamishra

1 Introduction

We study computational hardness in the setting of exact learning with membership queries (Angluin, 1988). In this setting, a learner has access to a membership query oracle and must exactly identify an unknown target concept. Given an input space X, we say that a concept class $C \subseteq \{0,1\}^X$ is efficiently learnable with membership queries alone if there is a learner that exactly identifies every unknown target $c \in C$ using queries and time polynomial in the relevant parameters.

A learning algorithm generally has two kinds of resources: (i) *information*, and (ii) *computation*. In the query learning setting, the learner obtains information about the target by querying an oracle. The number of queries needed for exact learning with membership queries is the *query complexity*; it is analogous to sample complexity in the more traditional setting of learning from random examples. The *computational complexity* is the amount of computation the learner needs in order to decide which inputs to query and how to process the labels it receives.

Query Complexity and Hardness. A concept class can be hard to learn simply because the learner must make many queries. For example, among Boolean functions over n variables, even elementary concept classes such as Singletons and general Conjunctions require exponentially many queries in the worst case. In such cases, the bottleneck is information-theoretic. As an example, consider the class of singletons $\mathsf{Sg} = \cup_{n\geq 1}\mathsf{Sg}_n$, which contains functions that output 1 on exactly one point $a \in \{0,1\}^n$ and output 0 everywhere else. The class Sg_n contains functions of the form $s_a : \{0,1\}^n \to \{0,1\}$. For each $a \in \{0,1\}^n$, the function $s_a \in \mathsf{Sg}_n$ is defined by

$$s_a(x) = \begin{cases} 1 & \text{if } x = a, \\ 0 & \text{otherwise.} \end{cases}$$

For the class Sg_n, it is straightforward to see that any exact query learner must make exponentially many queries in the worst case. To identify the target function, the learner must determine the hidden point $a \in \{0,1\}^n$ among 2^n possibilities. Regardless of the learning algorithm, an adversary can answer 0 to the first $2^n - 2$ membership queries. At that point, the learner still cannot rule out two possible singleton functions. To output the

target function correctly, the learner must make at least one more query. Hence, any exact learning algorithm for Sg_n must make $2^n - 1$ queries in the worst case.

Thus, some classes are hard to learn because they have large query complexity, and in such cases, it is often clear where the hardness arises: informative points are scarce. More generally, one can prove query lower bounds using combinatorial measures such as the logarithm of the size of a concept class, Teaching Dimension (Goldman and Kearns, 1995), and Extended Teaching Dimension (Hegedűs, 1995).

Computational Hardness. A natural question is whether there are concept classes for which a polynomial number of membership queries suffices, but no polynomial-time learner exists. In other words, can the bottleneck be computational rather than information-theoretic? If so, where and how can such computational bottlenecks arise in query learning? This is the main subject of this tutorial.

Result. Servedio and Gortler (2004, Thm. 8.2) answered this question positively by showing a concept class that cannot be learned in polynomial time using membership queries under the assumption that one-way functions exist.[1] In this tutorial, we give a slightly simpler proof of such a result. In particular, we construct a concept class C_n^{hw} that can be exactly learned with $\mathrm{poly}(n)$ queries by an algorithm running in exponential time (Prop. 3.1). However, assuming one-way functions exist, no polynomial-time learning algorithm can exactly learn C_n^{hw} (Sec. 3.3). We show how the same idea can be easily extended to construct a class that is computationally hard to query learn assuming NP $\not\subseteq$ RP (Sec. 3.4).

Approach. A query learning algorithm uses computation both to decide which inputs to query and to process the labels it receives. We construct a concept class C^{hw} with the following property: if there were an efficient query learning algorithm for C^{hw}, then that algorithm could be used to invert a one-way function. Since one-way functions are hard to invert on random inputs, this gives a contradiction.

Intuitively,[2] we construct a concept class in which each function has a hidden informative, or witness, point $x \in X$. For an unknown target $c^\star \in C^{\mathrm{hw}}$, if the learner knows this point, then it can query the label $c^\star(x)$ and identify the target function immediately. The challenge is to find this point. The class is designed so that the learner can easily make queries that reveal the information needed to identify the witness point. However, using that information to actually find the witness point is computationally hard: it requires inverting a one-way function or solving an NP-complete problem. The proof therefore illustrates how, even when the relevant information is available through a small number of queries, processing that information and determining the right query to make can still be computationally intractable.

The tutorial has two goals. First, it explains how a concept class can be computationally hard to learn with membership queries even when its query complexity is small. Second, it illustrates how cryptographic reductions and reductions from NP-complete problems can be used to prove computational hardness results in learning.

[1] They showed a separation with a concept class that can be learned in polynomial time by a quantum algorithm but not by a classical algorithm.

[2] This may not make a lot of sense at the minute.

LLMs. *The proof presented in Sec. 3.1-3.3 was largely developed by GPT 5.5 Pro* through a back-and-forth with the author, where the author did not contribute anything intellectually significant. The proof was verified and rewritten by the author to improve readability and provide additional context. The proof and construction in Sec. 3.4 were based on the author's observation that the construction can be easily extended to NP-complete problems. The broad goal was to give a digestible and simple proof of the existence of a concept class that is computationally hard to learn with queries alone. We believe the construction also intuitively illustrates how computational bottlenecks can arise in query learning.

2 Background and Preliminaries

2.1 Exact learning with membership queries

Let $\mathcal{X}$ be an instance space and let $C \subseteq \{0,1\}^{\mathcal{X}}$ be a concept class. In exact learning with membership queries, there is an unknown target concept $c^\star \in C$. The learning algorithm (or learner) has access to a membership query oracle $\mathrm{MQ} : \mathcal{X} \to \{0,1\}$. The learner may query any point $x \in X$, and receives the answer

$$\mathrm{MQ}(x) = c^\star(x) \in \{0,1\}.$$

The learner's goal is to output a hypothesis $h : X \to \{0,1\}$ such that

$$h(x) = c^\star(x) \qquad \text{for every } x \in \mathcal{X}.$$

In other words, the learner must recover the target concept exactly.

We say that a concept class C is *efficiently* exactly learnable if there is a (possibly probabilistic) polynomial-time algorithm which uses queries and time polynomial in the parameters of the concept class and for every $c^\star \in C$ returns a hypothesis h such that $h(x) = c^\star(x)$ for all $x \in \mathcal{X}$.

Complexity. We can assume that each call to the membership query oracle MQ takes $O(1)$ time. The minimum number of queries required by a learning algorithm to identify any target function in the class in the worst case is referred to as the query complexity of the class C. The time required by the learning algorithm to decide the inputs to query and how to process them determines the computational complexity.

2.2 Cryptographic background: one-way functions

For the result, we will use a central cryptographic assumption: the existence of one-way functions (OWFs). For convenience, we state the assumption in a length-regular form (Goldreich, 2001). Intuitively, OWFs are functions that are easy to compute but cannot be inverted in polynomial time for most inputs.

Definition 2.1 (Length-regular one-way function family)**.** A family of functions

$$f_n : \{0,1\}^n \to \{0,1\}^{m(n)}$$

is a length-regular one-way function family if the following conditions hold.

First, $m(n) \leq \text{poly}(n)$, and there is a deterministic polynomial-time algorithm which, given 1^n and $x \in \{0,1\}^n$, computes $f_n(x)$.

Second, f_n is hard to invert on a random input. More precisely, for every probabilistic polynomial-time algorithm $\mathcal{I}$,

$$\Pr_{x \sim \mathcal{U}(\{0,1\}^n)} \left[f_n\left(\mathcal{I}(1^n, f_n(x))\right) = f_n(x) \right] \leq \frac{1}{n^k}$$

for every constant $k \geq 1$ and for sufficiently large n. In other words, the quantity $\Pr_{x \sim \{0,1\}^n} \left[f_n\left(\mathcal{I}(1^n, f_n(x))\right) = f_n(x) \right]$ is smaller than all polynomials (also referred to as negligible) if $\mathcal{I}$ runs in polynomial time.

There are a couple of points to note here. First, the function f_n need not be injective (one-to-one), and hence the inverse $f^{-1}(f(x))$ is not necessarily unique. To invert $y = f_n(x)$, it is enough for the algorithm $\mathcal{I}$ to output any $\widehat{x} \in \{0,1\}^n$ satisfying

$$f_n(\widehat{x}) = y.$$

Secondly, this is an average-case statement. It says that no polynomial-time algorithm can find a preimage of $f_n(x)$ with non-negligible probability when x is chosen uniformly at random.

Remarks. The existence of one-way functions is one of the central conjectures of Cryptography. It is largely believed to be true at this moment [3] and many results rely on it, similar to the P $\neq$ NP conjecture. If P = NP then one-way functions cannot exist. If one shows that one-way functions exist, then it would imply that NP $\not\subset$ BPP (see Goldreich (2001) for more discussion). One candidate for a one-way function is the problem of factoring a large number $N = pq$. Clearly, given $p, q \in \mathbb{N}$, it is easy to check if $N = pq$, but given a large enough $N \in \mathbb{N}$, we do not have an efficient algorithm to find primes p, q.

3 A Computationally Hard to Learn Concept Class

We now discuss the concept class C^{hw} that is computationally hard to learn but has polynomial query complexity. We first describe the class and show how an algorithm that runs in exponential time can exactly learn the class with a polynomial number of queries. In Section 3.3, we discuss the main result, which shows that if one-way functions exist, then no polynomial-time algorithm can exactly learn the concept class C^{hw}.

3.1 The Concept Class C^{hw}

Pick a length-regular one-way function family

$$f_n : \{0,1\}^n \to \{0,1\}^{m(n)}.$$

[3] By very many researchers.

Instance Space. For a positive integer k, we use $[k]$ to denote $\{1, 2, \ldots, k\}$. The instance space $\mathcal{X}_n$ for our concept class has two parts:

$$\mathcal{X}_n = \{(0, i) : i \in [m(n)]\} \cup \{(1, x) : x \in \{0,1\}^n\}.$$

The first part will play the role of an address of some sort and has the same size as the output of our OWF f_n. The second part will play the role of a hidden witness.

Concept Class. Our concept class $C^{\text{hw}} = \cup_{n \geq 1} C_n^{\text{hw}}$ where C_n^{hw} is instantiated based on a one-way function f_n. The class C_n^{hw} contains functions from $\mathcal{X}_n$ to $\{0,1\}$. Each class C_n^{hw} has two types of functions: active functions act_y and null concepts nul_y for each $y \in \text{Im}(f_n)$.

For every string $y \in \{0,1\}^{m(n)}$, define the active concept $\mathsf{act}_y : \mathcal{X}_n \to \{0,1\}$ by

$$\mathsf{act}_y(0, i) = y_i \qquad \text{for every } i \in [m(n)],$$

and

$$\mathsf{act}_y(1, x) = \mathbf{1}[f_n(x) = y] \qquad \text{for every } x \in \{0,1\}^n.$$

Thus, for an OWF f_n, we have a function act_y for every $y \in \text{Im}(f_n)$ where act_y outputs y on the first part of the input space. In the second part, it outputs 1 only on the preimage of $y \in \text{Im}(f_n)$.

The null concepts are defined similarly, except they output 0 for all inputs in the second part $(1, x)$. For every string $y \in \{0,1\}^{m(n)}$, define the null concept $\mathsf{nul}_y : \mathcal{X}_n \to \{0,1\}$ by

$$\mathsf{nul}_y(0, i) = y_i \qquad \text{for every } i \in [m(n)],$$

and

$$\mathsf{nul}_y(1, x) = 0 \qquad \text{for every } x \in \{0,1\}^n.$$

Hence, the function nul_y writes the address y on the address part of the domain and is identically zero on the witness part.

The concept class is

$$C_n^{\text{hw}} = \{\mathsf{nul}_y : y \in \{0,1\}^{m(n)}\} \cup \{\mathsf{act}_y : y \in \text{Im}(f_n)\}.$$

Remark. Note that the instance space is designed that way for clarity. One can define the same instance space as Boolean vectors $\mathcal{X}_n = \{0,1\}^{1+m(n)+n}$. The same class C_n^{hw} can be defined over such bit strings where if the first bit is 0, the concepts are a restriction over $m(n)$ bits and if the first bit is 1, then they are a restriction over the last n bits. Hence, the same concept class can be defined with functions from $\{0,1\}^{1+m(n)+n}$ to $\{0,1\}$.

3.2 Polynomial Query Complexity

Before discussing why the class is computationally hard to learn, we first describe how an exponential-time algorithm can exactly learn the class with only a polynomial number of queries. The simple algorithm also illustrates where the computational bottleneck lies and why the class is designed in such a way.

Proposition 3.1. *There is a computationally unbounded learner which exactly identifies every target in C_n^{hw} using at most $m(n) + 1$ membership queries.*

Note that the learning algorithm has access to the one-way function f_n and it is not an unknown. The only unknown for the learner is the target function in the class and all concepts in the class C_n^{hw} are instantiated using the same function f_n.

Proof. Let $c^\star \in C_n^{\mathrm{hw}}$ be the unknown target. The learner first queries all address points

$$(0,1),(0,2),\ldots,(0,m(n))$$

and hence obtains the address string

$$y = \big(c^\star(0,1), c^\star(0,2), \ldots, c^\star(0,m(n))\big).$$

The learner then performs an exhaustive search over all $x \in \{0,1\}^n$ to determine whether there exists an x such that

$$f_n(x) = y.$$

If the algorithm finds such an x, then the learner queries the point $(1,x)$. If the query oracle returns 1, then the target is act_y. If it outputs 0, then it outputs nul_y.

If no such x exists, then $y \notin \mathrm{Im}(f_n)$, so the only concept in C_n with address y is nul_y. The learner outputs nul_y. Therefore, the learner exactly identifies the target using at most $m(n) + 1$ membership queries.

The running time may be exponential because of the exhaustive search for a preimage of $y \in \{0,1\}^{m(n)}$. □

Thus, from an information-theoretic perspective, the class is easy. Another way to show that the concept class has polynomial query complexity is to use the notion of Extended Teaching Dimension (Hegedűs, 1995). One can show that the Extended Teaching Dimension of C_n^{hw} is $O(m(n))$ and thus it also follows from the membership query Halving algorithm (Hegedűs, 1995) that an exponential time algorithm can exactly identify the class with $\mathrm{poly}(n)$ queries. The approach described above is more useful to conceptually understand how extra computation is more helpful and sufficient to exactly learn with a small number of queries.

3.3 Cryptographic Hardness of Polynomial-time Exact Identification

Proposition 3.1 shows that an exponential-time algorithm can learn the class C^{hw} using polynomial queries, but it does not rule out computationally efficient learners. In this section, we will describe the main result, which states that a polynomial-time learner cannot exist if we assume the existence of one-way functions.

Approach. In particular, we will prove it by contradiction. We argue that if there exists a probabilistic polynomial-time membership query algorithm for the class C^{hw}, then we can use the algorithm to invert the one-way function f_n in polynomial time with non-negligible probability. Thus, it directly contradicts the definition of one-way functions, and such algorithms cannot exist.

Theorem 3.2. *Assume that*

$$f_n : \{0,1\}^n \to \{0,1\}^{m(n)}$$

is a one-way function family. Then there is no probabilistic polynomial-time membership-query learner $\mathcal{A}$ such that, for every n and every target concept $c^\star \in C_n$,

$$\Pr\left[\mathcal{A}(\mathrm{MQ}_{c^\star}, 1^n) \text{ outputs a hypothesis exactly equivalent to } c^\star\right] \geq \frac{2}{3}.$$

The probability is over the internal randomness of $\mathcal{A}$.

Proof. Suppose, for contradiction, that such a learner $\mathcal{A}$ exists. We will construct a probabilistic polynomial-time inverter $\mathcal{I}$ for the one-way function family. The inverter $\mathcal{I}$ receives an input $(1^n, y)$, where y is intended to be of the form $f_n(x^\star)$ for a uniformly random $x^\star \in \{0,1\}^n$. The goal of $\mathcal{I}$ is to output some $x \in \{0,1\}^n$ satisfying

$$f_n(x) = y.$$

The inverter runs the learning algorithm $\mathcal{A}$ and simulates membership-query access to the active concept act_y. Importantly, this simulation does not require knowing a preimage of y. On address queries of the form $(0, i)$, the inverter answers y_i since it has access to y. On witness queries of the form $(1, x)$, the inverter computes $f_n(x)$ in polynomial time. If

$$f_n(x) = y,$$

then $\mathcal{I}$ has found a valid preimage of y, so it immediately outputs x. Otherwise, it answers the query with 0. This simulation is polynomial time because $\mathcal{A}$ runs in polynomial time and f_n is efficiently computable.

We now show that the inverter succeeds with noticeable probability. Fix any $y \in \mathrm{Im}(f_n)$. For this fixed y, consider two possible target concepts: nul_y and act_y. They have the same address part and differ only on witness points $(1, x)$ such that $f_n(x) = y$.

The set of null concepts C^{hw} helps us ensure that any algorithm that exactly learns C^{hw} must invert f_n. We will run two copies of $\mathcal{A}$ using the same random tape: one copy with membership-query access to nul_y, and the other with membership-query access to act_y. Simulating the MQ oracle nul_y is trivial. Let G_N be the event that the first copy succeeds in exactly identifying nul_y, and let G_A be the event that the second copy succeeds in exactly identifying act_y. Since we assume that $\mathcal{A}$ is correct, we have

$$\Pr[G_N] \geq \frac{2}{3} \qquad \text{and} \qquad \Pr[G_A] \geq \frac{2}{3}.$$

Thus, by the union bound,

$$\Pr[G_N \cap G_A] \geq \Pr[G_N] + \Pr[G_A] - 1 \geq \frac{1}{3}.$$

Let H_y be the event that, in the execution with oracle $\mathrm{MQ} = \mathsf{act}_y$, the learner queries some witness point $(1, x)$ satisfying

$$f_n(x) = y.$$

The oracles nul_y and act_y differ exactly on such points. Hence, if H_y does not occur, then the two coupled executions receive exactly the same answers to all their membership

queries. Consequently, they have the same transcript and produce the same output hypothesis.

But $\mathsf{nul}_y \neq \mathsf{act}_y$, since $y \in \mathrm{Im}(f_n)$. Therefore, the same output hypothesis cannot be exactly correct for both nul_y and act_y. It follows that $G_N \cap G_A \subseteq H_y$. Thus, the probability of the event H_y where the algorithm $\mathcal{A}$ makes a witness query $(1, x)$ satisfying $f_n(x) = y$ is

$$\Pr[H_y] \geq \frac{1}{3}.$$

Thus, when the inverter $\mathcal{I}$ simulates the learner with oracle $\mathrm{MQ} = \mathsf{act}_y$, it outputs a valid preimage of y exactly when the event H_y occurs. Therefore, for every $y \in \mathrm{Im}(f_n)$, we have

$$\Pr[\mathcal{I}(1^n, y) \text{ outputs } x \text{ with } f_n(x) = y] \geq \frac{1}{3}.$$

In particular, when $x^\star \sim \mathcal{U}(\{0,1\}^n)$ and $y = f_n(x^\star)$, we have

$$\Pr_{x^\star, I}\left[f_n\big(\mathcal{I}(1^n, f_n(x^\star))\big) = f_n(x^\star)\right] \geq \frac{1}{3}.$$

This is a non-negligible inversion probability, contradicting the one-wayness of f_n. Hence, no such polynomial-time exact membership-query learner $\mathcal{A}$ exists if we assume that one-way functions exist. □

3.4 Hardness based on NP-complete problems

The construction can be extended to NP-complete problems in a straightforward way. As an example, we show that one can obtain similar hardness results based on a reduction from the graph colouring problem. The key idea is that one part of the input domain can be used as the input for a computationally hard problem, and the other part is the set of possible answers. In the previous construction for C^{hw}, the first part was from the image of a one-way function where the computationally hard problem is to invert it. The second part was the input domain of the one-way function, which can be easily used to verify it.

For the graph colouring problem, we will encode the input graph in the first part of the domain. The rest of the domain contains the possible three colourings of the graph. Given a three-colouring of a graph, it is easy to check whether it is proper: no two adjacent vertices have the same colour. However, given a graph, finding a proper 3-colouring is NP-complete.

Recall that RP is the class of languages with randomised polynomial-time algorithms with one-sided error: no-instances are rejected with probability 1, and yes-instances are accepted with probability at least a fixed positive constant, say 2/3. We will use the assumption NP $\not\subseteq$ RP.

For a positive integer r, let $\mathcal{G}_r$ be the set of undirected graphs on vertex set $[r]$. Define the instance space

$$\mathcal{X}_r^{\mathrm{col}} = \{(0, e) : e \in \binom{[r]}{2}\} \cup \{(1, \chi) : \chi \in [3]^r\}.$$

The first part encodes the input graph. The second part contains all possible 3-colourings of the r vertices. Thus, the witness part is exponentially large, although each query point

has a polynomial-size description. As before, the input domain is described in this way for convenience, and it is straightforward to represent it as Boolean vectors.

For every graph $G \in \mathcal{G}_r$, define the null concept nul_G by

$$\mathsf{nul}_G(0, e) = \mathbf{1}[e \in E(G)] \qquad \text{and} \qquad \mathsf{nul}_G(1, \chi) = 0.$$

The active concept is defined using the verifier for graph colouring:

$$\mathsf{act}_G(0, e) = \mathbf{1}[e \in E(G)]$$

and

$$\mathsf{act}_G(1, \chi) = \mathbf{1}[\chi \text{ is a proper 3-colouring of } G].$$

Let

$$C_r^{\mathrm{col}} = \{\mathsf{nul}_G : G \in \mathcal{G}_r\} \cup \{\mathsf{act}_G : G \in \mathcal{G}_r\}, \qquad C^{\mathrm{col}} = \bigcup_{r \geq 1} C_r^{\mathrm{col}}.$$

If G is not 3-colourable, then $\mathsf{act}_G = \mathsf{nul}_G$. If G is 3-colourable, then the two concepts differ exactly on proper 3-colourings of G.

As before, C^{col} has polynomial query complexity with unbounded computation. A learner can query all points $(0, e)$, recover the graph G, and then search over all $\chi \in [3]^r$. If it finds a proper 3-colouring, it queries $(1, \chi)$, which distinguishes act_G from nul_G. If no such colouring exists, then the two concepts are identical. Hence at most $\binom{r}{2} + 1$ membership queries are sufficient.

Theorem 3.3. *Assume that* NP $\not\subseteq$ RP. *Then C^{col} is not exactly learnable by a probabilistic polynomial-time membership-query learner with success probability at least* $2/3$ *on every target concept.*

The proof follows the same line of arguments as Thm. 3.2.

Proof. Suppose, for contradiction, that such a learner $\mathcal{A}$ exists. We can use $\mathcal{A}$ to construct an RP algorithm for graph 3-colouring.

Given a graph $G \in \mathcal{G}_r$, run $\mathcal{A}$ while simulating membership queries to act_G. On a query $(0, e)$, answer $\mathbf{1}[e \in E(G)]$. On a query $(1, \chi)$, check in polynomial time whether χ is a proper 3-colouring of G. If it is, accept immediately; otherwise answer 0. If the learner terminates without querying a proper 3-colouring, reject.

If G is not 3-colourable, then the algorithm never accepts, so it has no false positives. Now suppose G is 3-colourable. Applying the same coupling argument as in Theorem 3.2 to the two targets act_G and nul_G, with the same random tape, shows that with probability at least $1/3$ the learner must query a point on which the two oracles differ. These points are precisely the proper 3-colourings of G. Therefore, the above algorithm accepts every 3-colourable graph with probability at least $1/3$.

By powering, we get that graph 3-colouring is in RP. Since graph 3-colouring problem is NP-complete, this implies NP $\subseteq$ RP, contradicting the assumption. Hence, no such polynomial-time exact membership-query learner exists. □

4 Discussion

The class C^{hw} is intended to be a simple example of computational hardness in query learning, much as Singletons are a simple example of hardness due to query complexity. Conceptually, the construction of C^{hw} is not far from the class of Singletons. In Singletons, the unknown target $c^\star$ is uniquely specified by a hidden point $x \in \{0,1\}^n$ such that $c^\star(x) = 1$. The learner's objective is to find this point. Since the learner has no useful information other than the answers to its queries, computation alone cannot help.

The classes C^{hw} and C^{col} have a similar structure. For an unknown target $c^\star \in C^{\mathrm{hw}}$, there is also a hidden witness point, or set of witness points, whose label can reveal the target function. However, the key difference is that, for C^{hw}, the learner does not lack information. After a polynomial number of queries, the learner can obtain enough information to determine what kind of witness point it should look for. The computational bottleneck is in using this information to actually find such a point. A computationally unbounded learner can do this by exhaustive search, but a polynomial-time learner cannot under standard complexity-theoretic assumptions.

The constructions in this tutorial are meant to show the existence of concept classes that are computationally hard to learn with membership queries, even though their query complexity is polynomial. However, it is unclear whether there are natural counterparts of these constructions that arise in practice. In particular, it is unclear whether there are concept classes of independent interest for which the main bottleneck in membership-query learning is computational rather than information-theoretic.

This question is better understood in related models of learning. The standard PAC-learning setting (Valiant, 1984) formalises learning from random examples, where the learner must produce a hypothesis with small error under the underlying data distribution. In this setting, several concept classes of interest, including DFAs and circuits, are computationally hard to learn even when polynomially many examples suffice. DFAs provide an interesting separation: learning from examples alone is computationally hard, while additional access to membership queries makes the learning problem feasible (Angluin, 1987). On a similar note, Angluin and Kharitonov (1991) showed that there are classes such as threshold circuits, DNFs, and CNFs, among others, that remain computationally hard to PAC-learn even when the learner has access to membership queries. Thus, in these settings, membership queries may provide additional information, but they do not necessarily remove the computational difficulty of learning.

References

Dana Angluin. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2):87–106, 1987.

Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

Dana Angluin and Michael Kharitonov. When won't membership queries help? In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 444–454, 1991.

Sally A Goldman and Michael J Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

Oded Goldreich. *Foundations of cryptography: volume 1, basic tools*, volume 1. Cambridge university press, 2001. ISBN 0-521-79172-3.

Tibor Hegedűs. Generalized teaching dimensions and the query complexity of learning. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 108–117, 1995.

Rocco A Servedio and Steven J Gortler. Equivalences and separations between quantum and classical learnability. *SIAM Journal on Computing*, 33(5):1067–1092, 2004.

Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.