The Complexity of Learning Concept Classes with Polynomial General Dimension*

Johannes Köbler and Wolfgang Lindner
Institut für Informatik
Humboldt-Universität zu Berlin
10099 Berlin, Germany

November 20, 2002

Abstract

We use the notion of general dimension to show that any p-evaluatable concept class with polynomial query complexity can be learned in polynomial time with the help of an oracle in the polynomial hierarchy, where the complexity of the required oracle depends on the query-types used by the learning algorithm. In particular, we show that for subset and superset queries an oracle in Σ_3^P suffices. Since the concept class of DNF formulas has polynomial query complexity with respect to subset and superset queries with DNF formulas as hypotheses, it follows that DNF formulas are properly learnable in polynomial time with subset and superset queries and the help of an oracle in Σ_3^{P}. We also show that the required oracle in our main theorem cannot be replaced by an oracle in a lower level of the polynomial-time hierarchy, unless the hierarchy collapses.

1 Introduction

In computational learning theory, one can distinguish between efficient learnability, which is usually modeled as learning in polynomial time, and polynomial query complexity, i.e. the possibility to learn a concept class with only a polynomial number of queries but unbounded computational resources. Clearly, polynomial-time learnability implies polynomial query complexity. On the other hand, in Angluin's query-learning model [2], it is known that for all combinations of equivalence and membership queries, polynomial query complexity implies polynomial-time learnability with additional access to an oracle in a low level of the polynomial-time hierarchy [6, 8, 11]. Thus, under the unlikely assumption that $\mathrm{P} = \mathrm{NP}$, polynomial query complexity in fact coincides with polynomial-time learnability

*Work supported by the DFG under project KO 1053/1-1

for equivalence and/or membership queries. There are, however, prominent examples such as boolean formulas, which can be learned with a polynomial number of equivalence queries, but there is high evidence that these concept classes cannot be learned in polynomial time (e.g., [9]).

Here we address the question whether similar results hold also for more powerful types of queries, such as subset and superset queries [2]. For equivalence and/or membership queries, the polynomial-time oracle algorithms in [6, 8, 11] are based on combinatorial characterizations of the corresponding polynomial query complexity. In [5], Balcázar et al. introduce the notion of the general dimension, a combinatorial measure which can be applied to arbitrary query-types and which characterizes, up to a logarithmic factor, the number of queries needed to learn a concept class. We use this notion to show, as our main result, that any p-evaluatable concept class with polynomial query complexity can be learned in polynomial time with the help of an oracle in the polynomial hierarchy, where the complexity of the required oracle depends on the query-types used by the learning algorithm. Similar as in [11] we use a modification of the majority-based algorithm of [5], where the emerging counting problems are solved by universal hashing techniques. Furthermore, our learning algorithm is proper in the sense that its output is a hypothesis from the concept class in question.

As a consequence, we get that all concept classes that are learnable with a polynomial number of equivalence and/or membership queries can be learned in polynomial time with an oracle in Σ_2^P, subsuming the results shown in [11]. A similar consequence holds also for subset and superset queries using an oracle in Σ_3^P. Since the concept class of DNF formulas has polynomial query complexity with respect to subset and superset queries with DNF formulas as hypotheses [4], it further follows that DNF formulas are properly learnable in polynomial time with subset and superset queries and the help of an oracle in Σ_3^P.

We further consider a concept class of [1] and show that this concept class is not learnable in polynomial time with an oracle in NP using equivalence queries with boolean circuits as hypotheses, unless the polynomial-time hierarchy collapses. This implies that the required oracle in our main theorem cannot be replaced by an oracle in a lower level of the polynomial-time hierarchy, unless the hierarchy collapses.

2 Preliminaries

We let $\mathcal{B}_n$ denote the set of all boolean functions $f : \{0,1\}^n \to \{0,1\}$. We assume the reader to be familiar with definitions and basic properties of the complexity classes in the polynomial-time hierarchy, as can be found in standard text books as, e.g., [12].

Let Σ be an alphabet. For a string $x \in \Sigma^*$, $|x|$ denotes its length. $\Sigma^{[n]}$ denotes the set of all strings $x \in \Sigma^*$ of length $|x| \leq n$. We assume the existence of a pairing function $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$ that is computable in polynomial time and has inverses also computable in polynomial time. $\langle \cdot, \cdot \rangle$ can be extended to encode finite sequences $(x_1, \ldots, x_k)$ of strings into a string $\langle x_1, \ldots, x_k \rangle \in \Sigma^*$. For a set A, $\|A\|$ denotes its cardinality.

Let C be a class of sets $A \subseteq \Sigma^*$. Then $\sharp$C denotes the class of functions $f : \Sigma^* \to \mathbb{N}$

such that there is a set $A \in \mathrm{C}$ and a polynomial p such that for all $x \in \Sigma^*$,

$$f(x) \;=\; \{y \in \Sigma^{p(|x|)} \mid \langle x, y\rangle \in A\}.$$

Let F be a class of functions $f : \Sigma^* \to \mathbb{N}$. Then $\max F$ denotes the class of functions $g : \Sigma^* \to \mathbb{N}$ such that there is a function $f \in F$ and a polynomial p such that for all $x \in \Sigma^*$,

$$g(x) \;=\; \max_{|y|=p(|x|)} f(\langle x, y\rangle).$$

The class $\min F$ ist defined analogously.

2.1 Learning Complexity and General Dimension

Balcázar et al. [5] introduced the general dimension of a boolean concept class to characterize the learning complexity with respect to arbitrary query protocols. The learning model presented in [5] is a generalization of the query learning model of Angluin [2]. Similar, but less general models have already been considered in [4, 13].

In the model of [5], learning of a concept class $\mathcal{C} \subseteq \mathcal{B}_n$ can be viewed as a game between a learning algorithm A and a teacher T with respect to some target $f \in \mathcal{C}$ that is only known to T. In each round, A asks a query q from a set $\mathcal{Q}$ of queries and T responds with a subset $\Lambda \subseteq \mathcal{B}_n$ that contains f. Thereby, T provides some partial knowledge about the target f in form of a *property* Λ shared by f. The communication between A and T is guided by some *protocol* $\mathcal{P} \subseteq \mathcal{Q} \times 2^{\mathcal{B}_n}$, i.e., the teacher is only allowed to respond with an answer Λ such that $\langle q, \Lambda\rangle \in \mathcal{P}$. The protocol $\mathcal{P}$ is required to be *complete* in the sense that for all $f \in \mathcal{B}_n$ and for all queries $q \in \mathcal{Q}$ there exists an answer $\Lambda \in \mathcal{P}_q^f$ where $\mathcal{P}_q^f = \{\Lambda \mid \langle q, \Lambda\rangle \in \mathcal{P} \wedge f \in \Lambda\}$.

For example, the protocol for equivalence queries with hypotheses from a concept class $\mathcal{C} \subseteq \mathcal{B}_n$ is the set of all pairs $(h, \{f \in \mathcal{B}_n \mid f(x) \neq h(x)\})$, for $h \in \mathcal{C}$ and $x \in \{0,1\}^n$, together with all pairs $(h, \{h\})$ with $h \in \mathcal{C}$. The first set of pairs corresponds to the case that a hypothesis h is answered by giving a counterexample x, and the second set of pairs corresponds to the answer “Yes”.

The goal of the learning algorithm is to collect enough knowledge about the target f such that f is the only remaining function in $\mathcal{C}$ that shares all properties exposed by T. More precisely, the *current version space* at any stage in the run of a learning algorithm A is the set of all functions in $\mathcal{C}$ that are contained in all answers received by A so far, and a concept class $\mathcal{C} \subseteq \mathcal{B}_n$ is *learnable* with at most d queries under a protocol $\mathcal{P}$, if there exists a learning algorithm A such that for all targets $f \in \mathcal{C}$ and any teacher T that answers each query q with some $\Lambda \in \mathcal{P}_q^f$, the only concept left in the current vesion space after at most d queries is f. The *learning complexity* of $\mathcal{C}$ under $\mathcal{P}$, denoted by $\mathrm{LC}(\mathcal{C}, \mathcal{P})$, is the smallest integer $d \geq 0$ such that $\mathcal{C}$ is learnable with at most d queries under $\mathcal{P}$. If no such integer d exists, then $\mathrm{LC}(\mathcal{C}, \mathcal{P}) = \infty$.

In order to characterize the learning complexity of a concept class $\mathcal{C}$ under an arbitrary protocol $\mathcal{P}$, Balcázar et al. introduce the *general dimension* of $\mathcal{C}$ under $\mathcal{P}$. The definition is based on the notion of an *answering scheme*, i.e., a subset $\mathcal{T} \subseteq \mathcal{P}$ such that for all queries $q \in \mathcal{Q}$ the set $\mathcal{T}_q = \{\Lambda \mid (q, \Lambda) \in \mathcal{T}\}$ is non-empty. Note that, in contrast to a protocol $\mathcal{P}$, an answering scheme $\mathcal{T}$ need not be complete since there might exist a query $q \in \mathcal{Q}$ and a function $f \in \mathcal{B}_n$ such that no answer $\Lambda \in \mathcal{T}_q$ contains f. The *general dimension* of $\mathcal{C}$ under $\mathcal{P}$, denoted by $\mathrm{Gdim}(\mathcal{C}, \mathcal{P})$, is the smallest integer $d \geq 0$ such that for all answering schemes $\mathcal{T} \subseteq \mathcal{P}$ there exists a set $\mathcal{S} \subseteq \bigcup_{q \in \mathcal{Q}} \mathcal{T}_q$ of cardinality at most d such that $\|\{f \in \mathcal{C} \mid (\forall \Lambda \in \mathcal{S})[f \in \Lambda]\}\| \leq 1$. If no such integer d exists, then $\mathrm{Gdim}(\mathcal{C}, \mathcal{P}) = \infty$.

It is shown in [5] that for each concept class $\mathcal{C}$ and protocol $\mathcal{P}$ it holds that $\mathrm{Gdim}(\mathcal{C}, \mathcal{P}) \leq \mathrm{LC}(\mathcal{C}, \mathcal{P}) \leq \mathrm{Gdim}(\mathcal{C}, \mathcal{P}) \lceil \ln \|\mathcal{C}\| \rceil$. Thus, the general dimension is in fact a combinatorial characterization of the learning complexity.

3 Polynomial Learning Complexity and Dimension

To define polynomial learning complexity and, in particular, polynomial-time learnability under an arbitrary protocol, we need to specify a way to represent concept classes and protocols. For concept classes, we use the following notations from [3] adapted to the boolean case.

Definition 1. *Let Σ and Γ be finite alphabets. A* representation of (boolean) concepts *is a set $C \subseteq 0^* \times \Gamma^* \times \Sigma^*$. With respect to any given $n \in \mathbb{N}$, we let $C_n = \{\langle u, x \rangle \mid \langle 0^n, u, x \rangle \in C\}$. The* concept *represented by a* concept name *$u \in \Gamma^*$ is $\kappa_{C_n}(u) = \{x \mid \langle u, x \rangle \in C_n\}$, and the* concept class *represented by C_n is $\mathcal{K}(C_n) = \{\kappa_{C_n}(u) \mid u \in \Gamma^*\}$.*

Here we always assume that $\Sigma = \{0, 1\}$. For the sake of notational brevity we simply write κ_n instead of κ_{C_n} whenever C is clear from the context. Furthermore, by abusing the notation, we identify the set $\kappa_n(u)$ with its characteristic function. Thus, we can view $\mathcal{K}(C_n)$ as a subset of $\mathcal{B}_n$.

The above definition allows us to regard a representation C of concepts as a decision problem. This means that we can express the usual assumption that the concept class represented by C can be evaluated in polynomial time by the fact that C is decidable in the complexity class P.

Example 2. *The circuit representation of boolean concepts, denoted by* Circ*, is the set of all tuples $\langle 0^n, c, x \rangle$ such that c is an encoding of a boolean circuit over the basis $\{\wedge, \vee, \neg\}$ with n input gates, x is a binary string of length n, and the circuit encoded by c accepts x.*

Now we define representations of protocols in a similar style as we defined represention of concepts. To illustrate the underlying idea let us reconsider the model of learning with equivalence queries with respect to a represention of concepts C. Here, a query is a concept name h, and the answer is either a counterexample in form of a binary string x, or the token "Yes". A counterexample x as an answer to some query h means that the target concept does not agree with the concept represented by h on x, i.e., the answer x to the query h means that the target is contained in the set of all concepts $\kappa_n(u)$ such that x is contained in the symmetric difference of $\kappa_n(u)$ and $\kappa_n(h)$. Similarly, the answer "Yes" to a query h

means that the target is contained in the singleton set $\{\kappa_n(h)\}$. Consequently, with respect to some fixed arity n, we represent a protocol $\mathcal{P}$ as a set P of triples $\langle 0^n, q, a, u\rangle$, where $q \in \Delta^*$ is a *query* and $a \in \Delta^*$ is an *answer*, for an additional finite alphabet Δ, and $u \in \Gamma^*$ is a concept name. An answer a together with a query q determine a set of concept names u satisfying $\langle 0^n, q, a, u\rangle \in P$, which, when interpreted with respect to a given represention of concepts C, describes the property associated with q and a.

Definition 3. *Let C be a representation of concepts, and let Δ be a finite alphabet. A* representation of a (boolean) protocol *with respect to C is a set $P \subseteq 0^* \times \Delta^* \times \Delta^* \times \Gamma^*$ which satisfies the following conditions for all $n \in \mathbb{N}$.*

1. *For all concept names u, and for all queries q, there exists an answer a such that $\langle 0^n, q, a, u\rangle \in P$.*

2. *For all concept names u and v, and for all queries q and answers a it holds that if $\langle 0^n, q, a, u\rangle \in P$ and $\kappa_n(u) = \kappa_n(v)$, then $\langle 0^n, q, a, v\rangle \in P$.*

With respect to any given integer $n \in \mathbb{N}$, we let $P_n = \{\langle q, a, u\rangle \mid \langle 0^n, q, a, u\rangle \in P\}$, and for a query q and an answer a we let $P_n(q, a) = \{u \mid \langle q, a, u\rangle \in P_n\}$. The property *associated with the pair $\langle q, a\rangle$ is $\Lambda_{P_n}(q, a) = \{\kappa_n(u) \mid u \in P_n(q, a)\}$ which we also denote by $\Lambda_n(q, a)$, and the* protocol *represented by P_n is $\mathcal{K}(P_n) = \{(q, \Lambda_n(q, a)) \mid q, a \in \Delta^*\}$.*

By the first condition, we have that $\mathcal{K}(P_n)$ is *complete with respect to* $\mathcal{K}(C_n)$ in the sense that for all $f \in \mathcal{K}(C_n)$ and for all queries $q \in \Delta^*$, there exists an answer $a \in \Delta^*$ with $f \in \Lambda_n(q, a)$. Clearly, completeness with respect to some proper subset $\mathcal{K}(C_n)$ of $\mathcal{B}_n$ is a strictly weaker condition than completeness with respect to $\mathcal{B}_n$ as required in [5]. It is, however, easy to see that the combinatorial characterization of [5] also holds if $\mathcal{K}(P_n)$ is only complete with respect to $\mathcal{K}(C_n)$.

The second condition is merely for the sake of notational convenience and implies that a concept $\kappa_n(u)$ is in the property $\Lambda_n(q, a)$ if and only if $\langle q, a, u\rangle \in P_n$.

Example 4. *Let $C \subseteq 0^* \times \Gamma^* \times \Sigma^*$ be a representation of concepts.*

- *The representation of the protocol for equivalence queries to C is the set* $\mathrm{Eq}(C) = \{\langle 0^n, h, x, u\rangle \mid h, u \in \Gamma^*, x \in \kappa_n(h) \triangle \kappa_n(u)\} \cup \{\langle 0^n, h, \text{“Yes”}, u\rangle \mid h, u \in \Gamma^*, \kappa_n(h) = \kappa_n(u)\}$.

- *The representation of the protocol for membership queries is the set* $\mathrm{Mem}(C) = \{\langle 0^n, x, \text{“Yes”}, u\rangle \mid u \in \Gamma^*, x \in \kappa_n(u)\} \cup \{\langle 0^n, x, \text{“No”}, u\rangle \mid u \in \Gamma^*, x \notin \kappa_n(u)\}$.

- *The representation of the protocol for subset queries to C is the set* $\mathrm{Sub}(C) = \{\langle 0^n, h, x, u\rangle \mid h, u \in \Gamma^*, x \in \kappa_n(h) \setminus \kappa_n(u)\} \cup \{\langle 0^n, h, \text{“Yes”}, u\rangle \mid h, u \in \Gamma^*, \kappa_n(h) \subseteq \kappa_n(u)\}$.

- *The representation of the protocol for subset and superset queries to C is the set* $\mathrm{Sub}(C) \oplus \mathrm{Sup}(C) = \{\langle 0^n, 0h, a, u\rangle \mid h, u \in \Gamma^*, \langle 0^n, h, a, u\rangle \in \mathrm{Sub}(C)\} \cup \{\langle 0^n, 1h, a, u\rangle \mid h, u \in \Gamma^*, \langle 0^n, h, a, u\rangle \in \mathrm{Sup}(C)\}$, *where* $\mathrm{Sup}(C)$ *is the representation for superset queries which is similarly defined as* $\mathrm{Sub}(C)$.

We now define polynomial learning complexity by imposing a polynomial bound both on the number of queries and the length of queries required for successful learning.

Definition 5. *Let C be a representation of concepts, and let P be a protocol representation with respect to C. Then C has* polynomial learning complexity *under P if there exist polynomials p and m, and an algorithm A which gets inputs s and n and may ask queries q of size at most $m(s, n)$, such that for all concept names u of size at most s, the following implication holds: If A always receives an answer a for each of its queries q satisfying $\kappa_n(u) \in \Lambda_n(q, a)$, then after at most $p(s, n)$ queries, A eventually halts and outputs a concept name h with $\kappa_n(h) = \kappa_n(u)$.*

In contrast to the definition of learning complexity, where the success condition is expressed in terms of the current version space, in the definition of polynomial learning complexity we require that a successful learning algorithm has to produce a concept name h for the target $\kappa_n(u)$. It is, however, easy to see that in the resource unbounded setting, both success conditions are equivalent.

Next we consider the corresponding notion of polynomial general dimension. We call a set $T \subseteq \Delta^{[m]} \times \Delta^*$ an *answering scheme for the length bound m*, if for each query q of length at most m there is an answer a with $\langle q, a\rangle \in T$. We further use $C_{s,n} = C_n \cap (\Gamma^{[s]} \times \Sigma^n)$ to denote the representation of concepts of size at most s.

Definition 6. *Let C be a representation of concepts, and let P be a protocol representation with respect to C. Then C has* polynomial general dimension *under P if there exist polynomials p and m, such that for all size bounds s, for all n, and for all answering schemes $T \subseteq \Delta^{[m(s,n)]} \times \Delta^*$ for the length bound $m(s, n)$ there exists a set $S \subseteq T$ of cardinality at most $p(s, n)$ such that $\|\{f \in \mathcal{K}(C_{s,n}) \mid (\forall \langle q, a\rangle \in S)[f \in \Lambda_n(q, a)]\}\| \leq 1$.*

Now we can use the arguments of [5] to show that polynomial learning complexity is equivalent to polynomial general dimension. The implication from polynomial general dimension to polynomial learning complexity is based on the fact that there always exists an inverse polynomially good query q for the current version space of any learning algorithm, where a good query q (with respect to C and P) is defined as follows.

Definition 7. *A query q is* δ-good *for a set of concepts $\mathcal{V}$ if each answer a to q eliminates at least a δ-fraction from $\mathcal{V}$, i.e. $\|\{f \in \mathcal{V} \mid f \notin \Lambda_n(q, a)\}\| \geq \delta\|\mathcal{V}\|$.*

Lemma 8 (cf. [5]). *Suppose that C has polynomial general dimension under P, and let p and m be the corresponding polynomials. Then, for all s and n, and for all non-empty sets $\mathcal{V} \subseteq \mathcal{K}(C_{s,n})$, there exists a query q of length at most $m(s, n)$ that is $(1 - 1/\|\mathcal{V}\|)/p(s, n)$-good for $\mathcal{V}$.*

By Lemma 8, for any set S of queries and answers received thus far, we can find a query q of polynomial length such that any answer to q eliminates at least an inverse polynomial fraction from the current version space $\mathcal{V} = \{f \in \mathcal{K}(C_{s,n}) \mid (\forall \langle q, a\rangle \in S)[f \in \Lambda_n(q, a)]\}$. Hence, after at most a polynomial number of queries, the only concept left in $\mathcal{V}$ is the target. This shows that polynomial general dimension implies polynomial learning complexity and we have the following equivalence.

Theorem 9. *Let C be a representation of concepts, and let P be a protocol representation with respect to C. Then the following are equivalent.*

1. *C has polynomial learning complexity under P.*

2. *C has polynomial general dimension under P.*

4 Polynomial-Time Learning with an Oracle in the Polynomial Hierarchy

In this section we show that any representation of concepts $C \in \mathrm{P}$ with polynomial general dimension under some representation P of a protocol can be learned in polynomial time with the help of an oracle whose complexity depends on the complexity of the decision problem P. We consider the following time-bounded analogue of polynomial learning complexity.

Definition 10. *A representation of concepts C is* polynomial-time learnable *under a protocol representation P if there is an algorithm A which fulfills all the conditions required in Definition 5, and whose running time is polynomially bounded in s and n.*

Obviously, any polynomial-time learning algorithm should be able to read the complete answer received at any stage. Thus, it is natural to require a polynomial bound on the possible answers in a protocol P.

Definition 11. *A representation of a protocol P is* polynomially (p-) honest *if there exists some polynomial l such that $|a| \leq l(n, |q|)$ for all $\langle q, a, u\rangle \in P_n$.*

As we will see below, if P can be decided in NP, then we get polynomial-time learnability with an oracle in Σ_2^P. In fact, we only need that the restriction of P specifying the properties $\Lambda_n(q, a)$ with $\|\Lambda_n(q, a)\| > 1$ can be decided in NP. This allows us to apply our theorem also to the important case of equivalence queries, where, in general, the part of P specifying "Yes" answers can only be decided in coNP.

Definition 12. *Let P be a protocol representation with respect to some representation of concepts C. An* admissible subset of P *is a set $P^* \subseteq P$ satisfying the following conditions for all n, q and a.*

1. *If $\|\Lambda_n(q, a)\| \neq 1$, then $P_n^*(q, a) = P_n(q, a)$.*

2. *If $\|\Lambda_n(q, a)\| = 1$, then $P_n^*(q, a) = P_n(q, a)$ or $P_n^*(q, a) = \emptyset$.*

Note that if $C \in \mathrm{P}$, then the protocol representation $\mathrm{Eq}(C)$ as given in Example 4 is p-honest. Moreover, $\mathrm{Eq}(C)$ is decidable in coNP and has the admissible subset $\{\langle 0^n, h, x, u\rangle \mid h, u \in \Gamma^*, x \in \kappa_n(h) \triangle \kappa_n(u)\}$ that is decidable in P.

Now we are ready to present our main theorem.

Theorem 13. *Let $C \in \mathrm{P}$ be a representation of concepts, and let $P \in \Sigma_2^P$ be a p-honest protocol representation with respect to C with an admissible subset $P^* \in \mathrm{NP}$. If C has polynomial general dimension under P, then C is polynomial-time learnable under P with an oracle in Σ_2^P.*

Before we proceed to the proof of Theorem 13, let us first discuss some consequences. By the remark above, it follows that for all representations $C \in \mathrm{P}$, C has polynomial learning complexity with respect to equivalence queries if and only if C is polynomial-time learnable with equivalence queries and an oracle in Σ_2^P. This holds also for equivalence and membership queries, and for membership queries alone. Thus, Theorem 13 subsumes all the results shown in [11].

Since Theorem 13 holds relative to any oracle A we get the following corollary.

Corollary 14. *Let $i \geq 1$, let $C \in \mathrm{P}$ be a representation of concepts, and let $P \in \Sigma_{i+1}^P$ be a p-honest protocol representation with respect to C with an admissible subset $P^* \in \Sigma_i^P$. If C has polynomial general dimension under P, then C is polynomial-time learnable under P with an oracle in Σ_{i+1}^P.*

For any $C \in \mathrm{P}$, the protocol representation $\mathrm{Sub}(C) \oplus \mathrm{Sup}(C)$ for subset and superset queries can be decided in $\Delta_2^P \subseteq \Sigma_2^P$. Hence, for all $C \in \mathrm{P}$, C has polynomial learning complexity with respect to subset and superset queries if and only if C is polynomial-time learnable with subset and superset queries and an oracle in Σ_3^P. Since the concept class of DNF-formulas can be learned with polynomial subset and superset queries (with DNF-formulas as hypotheses) [4], we get that this can be done also in polynomial time with an oracle in Σ_3^P.

Corollary 15. DNF *is polynomial-time learnable under* $\mathrm{Sub}(\mathrm{DNF}) \oplus \mathrm{Sup}(\mathrm{DNF})$ *relative to an oracle in* Σ_3^P.

4.1 Proof of Theorem 13

Let C be a representation of concepts with polynomial general dimension under a p-honest protocol representation $P \in \Sigma_2^P$ with an admissible subset $P^* \in \mathrm{NP}$. We have to show that C can be learned under P in polynomial time and with the help of an oracle in Σ_2^P.

Our algorithm will proceed similarly as the algorithm for Theorem 9 in the resource-unbouded setting. That is, for a given set S of queries and answers received thus far, we will try to find an inverse polynomially good query q such that any answer to q eliminates at least an inverse polynomial fraction from the current version space

$$V_{s,n}(S) = \{u \in \Gamma^{[s]} \mid (\forall \langle q, a\rangle \in S)[\langle q, a, u\rangle \in P_n]\}.$$

Then after a polynomial number of queries, the only concept left in $\mathcal{V}_{s,n}(S) = \{\kappa_n(u) \mid u \in V_{s,n}(S)$ is the target.

To compute such a good query q in polynomial time with an oracle in Σ_2^P, we will use well-known approximate counting techniques based on universal hashing, as already be done in [5, 11] for the specific case of equivalence queries. For this, however, we will have to consider the fraction of concept *names* rather than the fraction of concepts that are eliminated by any answer to q. That is, we will try to find an inverse polynomially good query q for the set $V_{s,n}(S)$ of concept names.

Definition 16. *A query q is* δ-good *for a set of concept names V if each answer a to q eliminates at least a δ-fraction from V, i.e. $\|\{u \in V \mid \langle q, a, u\rangle \notin P_n\}\| \geq \delta\|V\|$ for all $a \in \Delta^*$.*

Because the fraction of a set of concepts in $\mathcal{V}_{s,n}(S)$ might be very different from the fraction of the corresponding set of concept names in $V_{s,n}(S)$, we cannot use Lemma 8 directly to guarantee the existence of a sufficiently good query for V. For the specific case of equivalence queries, it is shown in [11] that an analogue of Lemma 8 also works for concept names rather than concepts. In the general case, however, the goodness of the query q depends on the maximal size of the equivalence classes $[u] = \{v \mid \kappa_n(v) = \kappa_n(u)\}$, $u \in V_{s,n}(S)$. We use the following notation.

Definition 17. *Let V be a finite set of concept names. The* weight *of a concept name u in V is $\mu(u) = \|[u] \cap V\|/\|V\|$. The* bias *of V is $\mu = \max_{u\in V} \mu(u)$.*

Now we can show the following analogue of Lemma 8 for concept names rather than concepts.

Lemma 18. *Suppose that C has polynomial general dimension under P via the polynomials p and m. Then, for all s and n, and for all non-empty sets $V \subseteq \Gamma^{[s]}$ with bias μ, there is a query q of length at most $m(s, n)$ that is $(1 - \mu)/p(s, n)$-good for V.*

Proof. Fix s and n, some non-empty set $V \subseteq \Gamma^{[s]}$, and assume that no query q of length at most $m(s, n)$ is $(1 - \mu)/p(s, n)$-good for V, i.e., for all queries q of length at most $m(s, n)$ there exists an answer a_q such that

$$\|\{u \in V \mid \langle q, a_q, u\rangle \notin P_n\}\| < \frac{\|V\| - \max_{u\in V} \|[u] \cap V\|}{p(s, n)}.$$

Consider the answering scheme $T = \{\langle q, a_q\rangle \mid q \in \Delta^{[m(s,n)]}\}$. Then, for any subset $S \subseteq T$ of cardinality at most $p(s, n)$, it follows that

$$\begin{aligned}\|\{u \in V \mid (\exists\langle q, a\rangle \in S)[\langle q, a, u\rangle \notin P_n]\}\| &\leq \sum_{\langle q,a\rangle\in S} \|\{u \in V \mid \langle q, a, u\rangle \notin P_n\}\| \\ &< \|V\| - \max_{u\in V} \|[u] \cap V\|,\end{aligned}$$

which implies that

$$\|\{u \in V \mid (\forall\langle q, a\rangle \in S)[\langle q, a, u\rangle \in P_n]\}\| > \max_{u\in V} \|[u] \cap V\|$$

and hence

$$\|\{\kappa_n(u) \mid u \in V \wedge (\forall\langle q, a\rangle \in S)[\langle q, a, u\rangle \in P_n]\}\| > 1.$$

Since the set of concepts $\{\kappa_n(u) \mid u \in V\}$ is contained in $\mathcal{K}(C_{s,n})$, this constradicts the assumption that C has polynomial general dimension under P. □ □

If the bias μ is not to large, i.e., $1-\mu$ is at least inverse polynomial, then Lemma 18 guarantees the existence of an inverse polynomially good query for V, and as we will see below, we then are able to compute such a query in polynomial time with an oracle in Σ_2^P. Thus, the remaining obstacle is to treat also the case when μ is large. This is the main departure from the special case of equivalence queries and the algorithm in the resource-unbouded setting.

However, if μ is large, then there exists a concept name u of large weight in V. Hence, if we replace the whole equivalence class $[u]$ with the single concept name u within V, then we eliminate a large fraction of concept names from V, and this does not effect the set of concepts $\mathcal{V}$ represented by V. To implement this idea, we maintain an additional set W of concept names, and we represent the current version space $\mathcal{V}_{s,n}(S)$ by the set

$$\begin{aligned} V_{s,n}(S,W) &= (V_{s,n}(S) \cap W) \cup (V_{s,n}(S) \setminus \bigcup_{w \in W} [w] \\ &= V_{s,n}(S) \cap \{u \in \Gamma^{[s]} \mid (\forall w \in W \setminus \{u\})[\kappa_n(u) \neq \kappa_n(w)]\} \end{aligned}$$

of concept names $u \in V_{s,n}(S)$ that are not equivalent to some $w \in W \setminus \{u\}$. This means, that for all concept names $u \in W$, $V_{s,n}(S,W)$ contains at most one concept name from the equivalence class $[u]$. By moving some u from V to W, we can discard $\|[u]\|-1$ elements from $V_{s,n}(S,W)$ without changing the set of concepts represented by $V_{s,n}(S,W)$.

Now we are ready to describe our algorithm. We initially set $S = W = \emptyset$. Then we repeat the following steps until $\|\mathcal{V}\| = 1$, and thus the current version space is reduced to the target. In each loop, we first compute an approximation $\tilde{\mu}$ of the bias μ satisfying $|\mu - \tilde{\mu}| \leq 1/12$. If $\tilde{\mu} \leq 2/3$, then $\mu \leq 2/3 + 1/12 = 3/4$, and hence, by Lemma 18, there exists a $(1/4p(s,n))$-good query. We then compute a $(1/6p(s,n))$-good query, and thus any answer to q eliminates at least a $(1/6p(s,n))$-fraction of concept names form V. If $\tilde{\mu} > 2/3$, then $\mu > 2/3 - 1/12 = 7/12$, and we proceed by computing a concept name u of weight $\mu(u) \geq 1/2$ in V. Note that $\mu > 7/12$ implies $\|V\| \geq 3$, and hence by adding u to W we eliminate at least $[u] - 1 \geq \|V\|/2 - \|V\|/3 = \|V\|/6$ concept names from V. Thus, in both cases, we eliminate at least a $1/6p(s,n)$-fraction of concept names form V, and it follows that after at most polynomially many queries, the only concept left in the current version space is the target. We then compute a concept name u in V and output u.

In the remainder of this section we have to show that each step in our learning algorithm can be done in polynomial time with an oracle in Σ_2^P.

First note that we only ask queries of length $m(s,n)$, and the honesty condition on P implies that all answers a we receive have length at most $l(n)$. Thus, the set S contains at most polynomially many pairs of queries and answers of polynomial length. Also note that the set W contains at most polynomially many concept names of length bounded by s.

To analyze the uniform comlexity of our algorithm, let $V^*_{s,n}(S,W)$ denotes the set $V_{s,n}(S,W)$ with the admissible subset $P^* \in \mathrm{NP}$ in place of $P \in \Sigma_2^P$, and consider the sets $V = \{\langle 0^s, 0^n, S, W, u\rangle \mid u \in V_{s,n}(S,W)\}$ and $V^* = \{\langle 0^s, 0^n, S, W, u\rangle \mid u \in V^*_{s,n}(S,W)\}$. Note that since $C \in \mathrm{P}$, V is in Σ_2^P and V^* is in NP. Further note that if $\|\mathcal{V}_{s,n}(S)\| > 1$, then

$V_{s,n}(S,W)$ coincides with $V^*_{s,n}(S,W)$. It follows that $\|\mathcal{V}_{s,n}(S)\| > 1$ if and only if $V^*_{s,n}(S,W)$ contains two concept names u and v with $\kappa_n(u) \neq \kappa_n(v)$. Thus, we can test whether there is more than one concept left in $\mathcal{V}_{s,n}(S,W)$ in polynomial time relative to an oracle in NP.

The final construction of a concept name $u \in V_{s,n}(S,W)$, when $\|\mathcal{V}_{s,n}(S)\| = 1$, can be done by prefix search in polynomial time with an oracle in Σ_2^P that contains all tuples $\langle 0^s, 0^n, S, W, u'\rangle$ such that there exists some u extending u' with $\langle 0^s, 0^n, S, W, u'\rangle \in V$.

For the remaining steps we use the already mentioned approximate counting techniques, which we summarize in the following two lemmas (cf. [10]).

Lemma 19. *Let $f \in \min\max\sharp\mathrm{NP}$. Then there exists an oracle $A \in \Sigma_2^P$ such that for all $x \in \Sigma^*$, and for all integers $a \geq 0$ and $e \geq 1$:*

1. $\langle x, a, 0^e\rangle \in A \Longrightarrow f(x) \leq (1 + \frac{1}{e})a$
2. $\langle x, a, 0^e\rangle \notin A \Longrightarrow f(x) > a.$

Lemma 20. *Let $f \in \min\max\sharp\mathrm{NP}$. Then there exists a function $\tilde{f} \in \mathrm{FP}(\Sigma_2^P)$ such that for all $x \in \Sigma^*$, and for all integers $e \geq 1$, $f(x) \leq \tilde{f}(x, 0^e) \leq (1 + \frac{1}{e})f(x)$.*

For all integers s, n, for all S and W, and for $\tau = \langle 0^s, 0^n, S, W\rangle$ let $t(\tau)$ be the cardinality of the set $V^*_{s,n}(S,W)$. Since V^* is in NP, $t(\tau)$ is in $\sharp$NP, and it follows immediately from Lemma 20 that we can compute an approximation $\tilde{t}(\tau, 0^e)$ of $t(\tau)$ in polynomial time and with an oracle in Σ_2^P, such that for all $e \geq 1$, $t(\tau) \leq \tilde{t}(\tau, 0^e) \leq (1 + \frac{1}{e})t(\tau)$.

Claim 21. *For all integers s, n and $e \geq 1$, for all S and W, and for $\tau = \langle 0^s, 0^n, S, W\rangle$ it is possible to compute an approximation $\tilde{\mu}(\tau, 0^e)$ of the bias $\mu(\tau)$ of $V^*_{s,n}(S,W)$ satisfying $|\tilde{\mu}(\tau, 0^e) - \mu(\tau)| \leq 1/e$ for all $e \geq 1$ in polynomial time with an oracle in Σ_2^P.*

Proof. Let $t(\tau)$ and $\tilde{t}(\tau, 0^e)$ as above, and consider the $\min\sharp$NP-function

$$b(\tau) = \min_{v \in \Gamma^{[s]}} \|V^*_{s,n}(S,W) \setminus [v]_n\|.$$

By Lemma 20 there is an approximation $\tilde{b}(\tau, 0^e)$ computable in polynomial time with an oracle in Σ_2^P such that $b(\tau) \leq \tilde{b}(\tau, 0^e) \leq (1 + \frac{1}{e})b(\tau)$.

Clearly, the bias $\mu(\tau)$ can be expressed as $\mu(\tau) = 1 - \frac{b(\tau)}{t(\tau)}$. Setting $\tilde{\mu}(\tau, 0^e) = 1 - \frac{\tilde{b}(\tau, 0^e)}{\tilde{t}(\tau, 0^e)}$, it follows that

$$\tilde{\mu}(\tau, 0^e) = \frac{\tilde{t}(\tau, 0^e) - \tilde{b}(\tau, 0^e)}{\tilde{t}(\tau, 0^e)} \leq \frac{t(\tau) + t(\tau)/e - b(\tau)}{t(\tau)} = \mu(\tau) + \frac{1}{e}$$

and since $b(\tau) \leq t(\tau)$ it also holds that

$$\tilde{\mu}(\tau, 0^e) \geq 1 - \frac{b(\tau) + b(\tau)/e}{t(\tau)} \geq \mu(\tau) - 1/e.$$

□ □

Now let us see how to construct a concept name of large weight in $V^*_{s,n}(S,W)$.

Claim 22. *For all integers s, n and $e \geq 1$, for all S and W, and for $\tau = \langle 0^s, 0^n, S, W\rangle$, it is possible to compute a concept name u of weight $\mu(\tau, u) \geq \mu(\tau) - 1/e$ in $V^*_{s,n}(S,W)$ in polynomial time with an oracle in Σ_2^P.*

Proof. Let $b(\tau)$ and $\tilde{b}(\tau, 0^e)$ as in the proof of the previous claim, and consider the $\sharp$NP-function

$$b(\tau, u) = \|V^*_{s,n}(S,W) \setminus [u]_n\|.$$

Applying Lemma 19 we get an oracle A in Σ_2^P such that for all integers $a \geq 0$ and $e \geq 1$, we have that

1. $\langle \tau, u, a, 0^e\rangle \in A \Longrightarrow b(u) \leq (1 + \frac{1}{e})a$
2. $\langle \tau, u, a, 0^e\rangle \notin A \Longrightarrow b(u) > a$

Let A' be the oracle consisting of all tuples $\langle \tau, u', a, 0^e\rangle$ for which there exists some $u \in \Gamma^{[s]}$ with prefix u' and $\langle \tau, u, a, 0^e\rangle \in A$. Since there exists some concept name u with $b(\tau, u) \leq b(\tau) \leq \tilde{b}(\tau, 0^e)$, and hence $\langle \tau, u, \tilde{b}, 0^e\rangle \in A$, it follows that we can construct a concept name u with $\langle \tau, u, \tilde{b}, 0^e\rangle \in A$ by prefix search in polynomial time and with the help of the oracle A' in Σ_2^P. Since for the resulting u it holds that $\langle \tau, u, \tilde{b}(\tau), 0^e\rangle \in A$, it follows that

$$b(\tau, u) \leq (1 + \frac{1}{e})\tilde{b}(\tau) \leq (1 + \frac{1}{e})^2 b(\tau) \leq (1 + \frac{3}{e})b(\tau)$$

and since $b(\tau) \leq t(\tau)$ we get

$$\mu(\tau, u) = 1 - \frac{b(\tau, u)}{t(\tau)} \geq 1 - \frac{b(\tau) + 3b(\tau)/e}{t(\tau)} \geq \mu(\tau) - \frac{3}{e}.$$

□ □

As our final application of Lemmas 19 and 20 we show how to construct a good query q.

Claim 23. *For all integers s, n and $e \geq 1$, for all S and W such that for $\tau = \langle 0^s, 0^n, S, W\rangle$ it holds that $\mu(\tau) \leq \frac{3}{4}$, it is possible to compute a query q of length $m(s,n)$ that is $(\frac{1}{4p(s,n)} - 1/e)$-good for $V^*_{s,n}(S,W)$ in polynomial time with an oracle in Σ_2^P.*

Proof. Consider the max $\sharp$NP-function

$$c(\tau, q) = \max_{a \in \Delta^{[l(n,|q|)]}} \|\{u \in V^*_{s,n}(S,W) \mid \langle q, a, u\rangle \in P^*_n\}\|$$

and let A be the oracle in Σ_2^P provided by Lemma 19 such that for all integers $a \geq 0$ and $e \geq 1$, we have that

1. $\langle \tau, q, a, 0^e\rangle \in A \Longrightarrow c(\tau, q) \leq (1 + \frac{1}{e})a$,

2. $\langle \tau, q, a, 0^e\rangle \notin A \Longrightarrow c(\tau, q) > a$.

By Lemma 18, the assumption $\mu \leq \frac{3}{4}$ implies that there exists a query q of length $m(s, n)$ such that q is $1/4p(s, n)$-good for $V^*_{s,n}(S, W)$. This means that for all answers a it holds that

$$\begin{aligned}\|\{u \in V^*_{s,n}(S, W) \mid \langle q, a, u\rangle \in P^*_n\}\| &\leq \|\{u \in V^*_{s,n}(S, W) \mid \langle q, a, u\rangle \in P_n\}\| \\ &\leq (1 - \frac{1}{4p(s, n)})t(\tau)\end{aligned}$$

and hence $c(\tau, q) \leq (1 - 1/4p(s, n))t(\tau) \leq (1 - 1/4p(s, n))\tilde{t}(\tau, 0^e)$. Setting $a = (1 - 1/4p(s, n))\tilde{t}(\tau, 0^e)$ it follows that $\langle \tau, q, a, 0^e\rangle \in A$, and similarly as in the previous claim we can construct query q in polynomial time and with the help of an oracle in Σ^P_2 such that $\langle \tau, q, a, 0^e\rangle \in A$. Now, $\langle \tau, q, a, 0^e\rangle \in A$ implies

$$\begin{aligned}c(\tau, q) \leq (1 + \frac{1}{e})a &= (1 + \frac{1}{e})(1 - \frac{1}{4p(s, n)})\tilde{t}(\tau, 0^e) \leq (1 + \frac{1}{e})^2(1 - \frac{1}{4p(s, n)})t(\tau) \\ &\leq (1 - \frac{1}{4p(s, n)} + \frac{3}{e})t(\tau).\end{aligned}$$

Since $c(\tau, q)$ is defined in terms of P^* rather than P, this only means that for all answers a with $\|\Lambda_n(q, a)\| > 1$ it holds that

$$\begin{aligned}\|\{u \in V^*_{s,n}(S, W) \mid \langle q, a, u\rangle \in P_n\}\| &= \|\{u \in V^*_{s,n}(S, W) \mid \langle q, a, u\rangle \in P^*_n\}\| \\ &\leq (1 - \frac{1}{4p} + \frac{3}{e})t(\tau).\end{aligned}$$

The assumption $\mu \leq \frac{3}{4}$ implies that also for all answers a with $\|\Lambda_n(q, a)\| = 1$ it holds that $\|\{u \in V^* \mid \langle q, a, u\rangle \in P_n\}\| \leq \mu(\tau)t(\tau) \leq (\frac{3}{4})t(\tau)$. ☐ ☐

5 Non-learnability with an oracle in NP

In this section we show that the Σ^P_2 oracle in our main theorem cannot be replaced by an oracle in NP, unless the polynomial hierarchy collapses to Δ^P_2. This computational hardness result is based on the *representation problem* REP(C) [14, 1] for a representation C,

$$\mathrm{REP}(C) = \{\langle 0^s, 0^n, c\rangle \mid (\exists u \in \Gamma^{[s]})[\kappa_{C_n}(u) = \kappa_{\mathrm{Circ_n}}(c)]\},$$

where Circ is the circuit representation for boolean functions.

Aizenstein et al. [1] showed that there is a representation $K \in \mathrm{P}$ such that its representation problem REP(K) is complete for Σ^P_2. The representation K can be described as follows. Concept names are of the form $c \vee t$, where c is a boolean circuit over an even number of variables $x_1, \ldots, x_n$, and t is a conjunction containing exactly one literal for each of the first $n/2$ variables $x_1, \ldots, x_{n/2}$. The concept represented by $c \vee t$ is the set of all $x \in \{0,1\}^n$ such that c accepts x or t is true under the assignment x. (In [1] a 3CNF formula is used in place of a circuit as above. This, however, does not affect the Σ_2^P-completeness of REP(K).)

Let us first consider the complexity of learning K with equivalence queries to K. By answering each query $c \vee t$ with a counterexample x that fulfills the conjunction t it is easy to see that any learner under Eq(K) needs at least $2^{n/2} - 1$ queries to identify the target.

Proposition 24. *K does not have polynomial learning complexity under* Eq(K).

If we allow arbitrary circuits as hypotheses, then in contrast to the previous proposition, K is learnable with a polynomial number of queries by using a simple majority-vote strategy.

Proposition 25. *K has polynomial learning complexity under* Eq(Circ).

Recall that in our definition of polynomial learning complexity for a representation of concepts C, we insist that the learning algorithm A outputs a concept name h such that independently of the protocol, the concept represented by h with respect to C is equivalent to the target. This allows us to proof that if K is polynomial-time learnable under Eq(Circ) with an oracle in NP, then the representation problem for K is in Δ_2^P. Combining this with the Σ_2^P-completeness of REP(K) we get the following theorem.

Theorem 26. *K is not polynomial-time learnable under* Eq$(Circ)$ *with an oracle in* NP*, unless the polynomial-time hierarchy collapses to Δ_2^P.*

Thus we have found representations C and P satisfying the conditions of Theorem 13 but C is not polynomial-time learnable under P relative to an oracle in NP, unless the polynomial hierarchy collapses to Δ_2^P. In fact, by Theorem 13, K is polynomial-time learnable under $Eq(Circ)$ with an oracle in NP if and only if the polynomial-time hierarchy collapses to Δ_2^P.

The non-learnability of K under $Eq(Circ)$ with an oracle in NP relies on the fact, that we allow equivalence queries that are arbitrary circuits but we insist that the output is of the form $c \vee t$. For a similar non-learnability result for K where both the output and the hypothesis are of the form $c \vee t$, we now consider learning with *projective* equivalence queries [7, 4]. A projective equivalence query with respect to a representation C of hypotheses is a pair of the form $\langle \alpha, h \rangle$, where $\alpha \in \{0,1,*\}^n$. The *partial assignment* α describes the hypercube G_α consisting of all strings $x \in \{0,1\}^n$ such that x coincides with α on all positions where α is not $*$. In response to a query $\langle \alpha, h \rangle$, the answer is "Yes" if $\kappa_n(h)$ coincides with the target on all strings in the hypercube G_α. Otherwise, the answer consists of a string x in the hypercube G_α for which $\kappa_n(h)$ does not agree with the target. Let Proj-Eq(C) denote a representation of the corresponding protocol. By adapting the proof in [11] that boolean circuits are polynomial-time learnable under Eq$(Circ)$ with an oracle in Σ_2^P we can show the following theorem.

Theorem 27. *K is polynomial-time learnable under* Proj-Eq(K) *relative to an oracle in Σ_2^P.*

Similarly as in Theorem 26, the learnability of K under Proj-Eq(K) with an oracle in NP implies that the representation problem from K is in Δ_2^P.

Theorem 28. *K is not polynomial-time learnable under* Proj-Eq(K) *relative to an oracle in* NP*, unless the polynomial-time hierarchy collapses to Δ_2^P.*

6 Acknowledgements

For helpful conversations and suggestions on this work we are very grateful to José Balcázar, Jorge Castro, Richard Gavaldà, and David Guijarro.

References

[1] H. Aizenstein, T. Hegedüs, L. Hellerstein, and L. Pitt. Complexity theoretic hardness results for query learning. *Computational Complexity*, 7:19–53, 1998.

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

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

[4] J. Balcázar, J. Castro, and D. Guijarro. Abstract combinatorial characterizations of exact learning via queries. In *Proc. 13th ACM Conference on Computational Learning Theory*, pages 248–254. Morgan Kaufmann, 2000.

[5] J. Balcázar, J. Castro, and D. Guijarro. A general dimension for exact learning. In *Proc. 14th ACM Conference on Computational Learning Theory*, volume 2111 of *Lecture Notes in Artificial Intelligence*, pages 354–367. Springer-Verlag, Berlin Heidelberg New York, 2001.

[6] N. 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.

[7] L. Hellerstein and M. Karpinsky. Learning read-once formulas using membership queries. In *Proc. 2nd ACM Conference on Computational Learning Theory*, pages 146–161. Morgan Kaufmann, 1989.

[8] L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan, and D. Wilkins. How many queries are needed to learn? *Journal of the ACM*, 43(5):840–862, 1996.

[9] M. J. Kearns and L. G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41:67–95, 1994.

[10] J. Köbler. Lowness-Eigenschaften und Erlernbarkeit von Booleschen Schaltkreisklassen. Habilitationsschrift, Universität Ulm, 1995.

[11] J. Köbler and W. Lindner. Oracles in Σ_2^p are sufficient for exact learning. *International Journal of Foundations of Computer Science*, 11(4):615–632, 2000.

[12] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[13] O. Watanabe. A framework for polynomial time query learnability. *Mathematical Systems Theory*, 27:211–229, 1994.

[14] O. Watanabe and R. Gavaldà. Structural analysis of polynomial time query learnability. *Mathematical Systems Theory*, 27:231–256, 1994.