The Complexity of Learning Concept Classes with Polynomial General Dimension [⋆]

Johannes Köbler [a], Wolfgang Lindner [b]

[a] *Institut für Informatik, Humboldt-Universität zu Berlin, 10099 Berlin, Germany*

[b] *Abteilung Theoretische Informatik, Universität Ulm, 89069 Ulm, Germany*

Abstract

The general dimension is a combinatorial measure that characterizes the number of queries needed to learn a concept class. We use this notion 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.

Key words: query learning, learning complexity, learning DNF formulas, polynomial-time hierarchy

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.

[⋆] Work supported by the DFG under project KO 1053/1-1

On the other hand, in Angluin's query-learning model (1), 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 (2; 3; 4). Thus, under the unlikely assumption that P = NP, polynomial query complexity in fact coincides with polynomial-time learnability 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., (5)).

Here we address the question whether similar results hold also for more powerful types of queries, such as subset and superset queries (1). For equivalence and/or membership queries, the polynomial-time oracle algorithms in (2; 3; 4) are based on combinatorial characterizations of the corresponding polynomial query complexity.

In (6), Balcázar et al. introduce the notion of a 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. As in (4) we use a modification of the majority-based algorithm of (6), 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 (4). 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 (7), it further follows that DNF formulas are properly learnable in polynomial time with subset and superset queries and the help of a Σ_3^P oracle.

We further consider a particular concept class of (8) 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. A similar result is shown for the case of (proper) projective equivalence queries. These results 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.

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., (9).

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 $\sharp\mathrm{C}$ 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 $x \in \Sigma^*$,

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

The class $\min F$ is defined analogously.

2.1 Learning Complexity and General Dimension

Balcázar et al. (6) 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 (6) is a generalization of the query learning model of Angluin (1). Similar, but less general models have already been considered in (7; 10).

In the model of (6), 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 w.r.t.* $\mathcal{B}_n$ 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 version 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 (6) 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 (11) 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 polynomial time.

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 represenions 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 quadruples $\langle 0^n, q, a, u\rangle$, 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 $u \in \Gamma^*$, and for all queries $q \in \Delta^*$, there exists an answer $a \in \Delta^*$ such that $\langle 0^n, q, a, u\rangle \in P$.
(2) For all concept names $u, v \in \Gamma^*$, and for all queries $q \in \Delta^*$ and answers $a \in \Delta^*$ 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 (6). It is, however, easy to see that the combinatorial characterization of (6) 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 means that P is *semantically closed.* It implies that a concept $\kappa_n(u)$ has the property $\Lambda_n(q, a)$ if and only if $\langle q, a, u\rangle \in P_n$. Also we note that a protocol representation P should not be confused with the complexity class P.

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

(1) The representation of the protocol for equivalence queries to C is the set

$$\begin{aligned}\mathrm{Eq}(C) = &\{\langle 0^n, h, x, u\rangle \mid h, u \in \Gamma^*, x \in \kappa_n(h) \triangle \kappa_n(u)\}\\ &\cup \{\langle 0^n, h, \text{“Yes”}, u\rangle \mid h, u \in \Gamma^*, \kappa_n(h) = \kappa_n(u)\}\end{aligned}$$

This means that for a query h the answer x gives the information that the target belongs to the set $\Lambda_n(h, x) = \{\kappa_n(u) \mid u \in \Gamma^*, x \in \kappa_n(h) \triangle \kappa_n(u)\} = \{c \in \mathcal{K}(C_n) \mid x \in \kappa_n(h) \triangle c\}$.

(2) The representation of the protocol for membership queries is the set

$$\begin{aligned}\mathrm{Mem}(C) = &\{\langle 0^n, x, \text{“Yes”}, u\rangle \mid u \in \Gamma^*, x \in \kappa_n(u)\}\\ &\cup \{\langle 0^n, x, \text{“No”}, u\rangle \mid u \in \Gamma^*, x \notin \kappa_n(u)\}.\end{aligned}$$

(3) The representation of the protocol for subset queries to C is the set

$$\begin{aligned}\mathrm{Sub}(C) = &\{\langle 0^n, h, x, u\rangle \mid h, u \in \Gamma^*, x \in \kappa_n(h) \setminus \kappa_n(u)\}\\ &\cup \{\langle 0^n, h, \text{“Yes”}, u\rangle \mid h, u \in \Gamma^*, \kappa_n(h) \subseteq \kappa_n(u)\}.\end{aligned}$$

(4) The representation of the protocol for subset and superset queries to C is the set

$$\begin{aligned}\mathrm{Sub}(C) \oplus \mathrm{Sup}(C) = &\{\langle 0^n, 0h, a, u\rangle \mid h, u \in \Gamma^*, \langle 0^n, h, a, u\rangle \in \mathrm{Sub}(C)\}\\ &\cup \{\langle 0^n, 1h, a, u\rangle \mid h, u \in \Gamma^*, \langle 0^n, h, a, u\rangle \in \mathrm{Sup}(C)\},\end{aligned}$$

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 in C_n 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 (6) 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. (6)) *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 $\mathcal{V}$.*

PROOF. Fix s and n, some non-empty set $\mathcal{V} \subseteq \mathcal{K}(C_{s,n})$, and assume that no query q of length at most $m(s,n)$ is $(1 - 1/\|\mathcal{V}\|)/p(s,n)$-good for $\mathcal{V}$, i.e., for all queries q of length at most $m(s,n)$ there exists an answer a_q such that $\|\{f \in \mathcal{V} \mid f \notin \Lambda_n(q,a)\}\| < (\|\mathcal{V}\| - 1)/p(s,n)$. 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

$$\begin{aligned}\|\{f \in \mathcal{V} \mid (\exists \langle q,a\rangle \in S)[f \notin \Lambda_n(q,a)]\}\| &\leq \sum_{\langle q,a\rangle \in S} \|\{f \in \mathcal{V} \mid f \notin \Lambda_n(q,a)\}\| \\ &< \|\mathcal{V}\| - 1 \ ,\end{aligned}$$

which implies that $\|\{f \in \mathcal{V} \mid (\forall \langle q,a\rangle \in S)[f \in \Lambda_n(q,a)]\}\| > 1$. Since $\mathcal{V} \subseteq \mathcal{K}(C_{s,n})$, this contradicts the assumption that C has polynomial general dimension under P via p and m. $\square$

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 length bound on the possible answers in a protocol P.

Definition 11 *A representation of a protocol P is* polynomially honest *(*p-honest *for short) 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. Intuitively, we can drop this part from P since the learning algorithm has already finished its task as soon as it receives an answer a to a query q with $\|\Lambda_n(q, a)\| = 1$.

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 $P_n^*(q, a) = \emptyset$.

protocol P	complexity of P	complexity of an admissible subset P^*	oracle complexity
$Mem(C)$	P	P	Σ_2^P
$Eq(C)$	coNP	P	Σ_2^P
$Mem(C) \oplus Eq(C)$	coNP	P	Σ_2^P
$Sub(C)$	coNP	coNP	Σ_3^P
$Sup(C)$	coNP	coNP	Σ_3^P
$Sub(C) \oplus Sup(C)$	coNP	coNP	Σ_3^P

Fig. 1. Upper bounds for the oracle complexity of learning algorithms

Example 13 *Provided that $C \in \mathrm{P}$, the protocol representation* Eq(C) *as given in Example 4 is decidable in* coNP *and has the admissible subset $P^* = \{\langle 0^n, h, x, u\rangle \mid h, u \in \Gamma^*, x \in \kappa_n(h) \triangle \kappa_n(u)\}$ that is decidable in* P*.*

Now we are ready to present our main theorem.

Theorem 14 *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 14, 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 14 subsumes all the results shown in (4).

The table in Figure 1 summarizes our upper bounds for the oracle complexity of polynomial-time learning algorithms for various protocols P with respect to concept representations $C \in \mathrm{P}$.

Since the proof of Theorem 14 given below relativizes to an arbitrary oracle we get the following corollary.

Corollary 15 *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 $\mathrm{coNP} \subseteq \Sigma_2^P$. 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) (7), we get that this can be done also in polynomial time with an oracle in Σ_3^P.

Corollary 16 DNF *is polynomial-time learnable under the protocol representation* $\mathrm{Sub}(\mathrm{DNF}) \oplus \mathrm{Sup}(\mathrm{DNF})$ *with an oracle in Σ_3^P.*

The rest of this section is devoted to the proof of Theorem 14.

4.1 Good Queries for Sets of Concept Names

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 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 a good query q such that any answer to q eliminates at least an inverse polynomial fraction from the set of concept names representing 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 the current version space $\mathcal{V}_{s,n}(S) = \{\kappa_n(u) \mid u \in V_{s,n}(S)\}$ is the target.

To compute such a good query q in polynomial time with an oracle in Σ_2^P, we will apply well-known approximate counting techniques based on universal hashing that have also been used in (6; 4) 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 the answers of the teacher. That is, the algorithm needs to find queries that are good for the set $V_{s,n}(S)$ of concept names.

Definition 17 *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 (4) 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 $[u] = \{v \mid \kappa_n(v) = \kappa_n(u)\}$, $u \in V_{s,n}(S)$. To be more precise, we introduce the following notation.

Definition 18 *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 19 *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

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

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. □

4.2 The Algorithm

As usual, the current knowledge of the learning algorithm is represented by the set S of queries and answers received thus far. If the bias μ of $V_{s,n}(S)$ is not too large, i.e., $1-\mu$ is at least inverse polynomial, then Lemma 19 guarantees the existence of an inverse polynomially good query for $V_{s,n}(S)$, 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. In this case, our algorithm significantly differs from the algorithm used in the proof of Theorem 9 in the resource-unbounded setting.

If μ is large, then there exists a concept name u of large weight in $V_{s,n}(S)$. Hence, if we replace the whole equivalence class $[u]$ with the single concept name u within $V_{s,n}(S)$, then we eliminate a large fraction of concept names from $V_{s,n}(S)$, and this does not affect the set of concepts $\mathcal{V}_{s,n}(S)$ represented by $V_{s,n}(S)$. To implement this idea, we maintain an additional set W of pairwise non-equivalent concept names, and we represent the current version space $\mathcal{V}_{s,n}(S)$ by the set $V_{s,n}(S,W)$ which contains only those concept names in $V_{s,n}(S)$ that are not equivalent to some $w \in W$ or itself belong to W, i.e.,

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

Thus, for all concept names $u \in W$, $V_{s,n}(S,W)$ contains at most one concept name from the equivalence class $[u]$. Note that, by including some u from $V_{s,n}(S,W)\setminus\bigcup_{w\in W}[w]$ into W, we discard $\|[u]\cap V_{s,n}(S,W)\|-1$ elements from $V_{s,n}(S,W)$ without changing the set $\mathcal{V}_{s,n}(S)$ of concepts represented by $V_{s,n}(S,W)$.

Now we are ready to describe our algorithm (cf. Figure 2). We initially set $S = W = \emptyset$. Then we repeat the following steps until $\|\mathcal{V}_{s,n}(S)\| = 1$, 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 of $V_{s,n}(S,W)$ satisfying $|\mu-\tilde{\mu}| \leq 1/12$. If $\tilde{\mu} \leq 2/3$, then for the actual bias μ of $V_{s,n}(S,W)$ it holds that $\mu \leq 2/3+1/12 = 3/4$, and hence, by Lemma 19, there exists a $(1/4p(s,n))$-good query for $V_{s,n}(S,W)$. We then compute a $(1/6p(s,n))$-good query q, and thus any answer to q eliminates at least a $(1/6p(s,n))$-fraction of concept names form $V_{s,n}(S,W)$. If on the other hand, $\tilde{\mu} > 2/3$, 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 $V_{s,n}(S,W)$. Note that $\mu > 7/12$ implies $\|V_{s,n}(S,W)\| \geq 3$, and hence by adding u to W we eliminate at least $\|[u]\cap V_{s,n}(S,W)\| - 1 \geq \|V_{s,n}(S,W)\|/2 - \|V_{s,n}(S,W)\|/3 = \|V_{s,n}(S,W)\|/6$ concept names from $V_{s,n}(S,W)$. Thus, in both cases, we eliminate at least a $1/6p(s,n)$-fraction of concept names form $V_{s,n}(S,W)$, and it follows that after at most polynomially many loops, the

```
$S \leftarrow \emptyset$; $W \leftarrow \emptyset$
while $\|\mathcal{V}_{s,n}(S)\| > 1$ do
  compute an approximation $\tilde{\mu}$ of the bias $\mu$ of $V_{s,n}(S, W)$
  satisfying $|\mu - \tilde{\mu}| \le 1/12$
  if $\tilde{\mu} \le 2/3$ then
    compute a $(1/6p(s, n))$-good query for $V_{s,n}(S, W)$
    ask $q$ and receive an answer $a$
    $S \leftarrow S \cup \{(q, a)\}$
  else
    compute a concept name $u$ of weight $\mu(u) \ge 1/2$ in $V_{s,n}(S, W)$
    $W \leftarrow W \cup \{u\}$
  end if
end while
compute a concept name $u$ in $V_{s,n}(S, W)$ and output $u$
```

Fig. 2. The algorithm for proving Theorem 14

only concept left in the current version space is the target. We then compute a concept name u in $V_{s,n}(S, W)$ and output u.

4.3 The complexity of the Algorithm

To complete the proof of Theorem 14 it only remains to show that each step of 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 size of S as well as the length of its elements grow at most polynomially. Also note that since the algorithm performs at most polynomially many loop iterations, the set W contains at most a polynomial number of concept names of length bounded by s.

To analyze the uniform complexity of our algorithm, let $V_{s,n}^*(S, W)$ denote the subset of $V_{s,n}(S, W)$ that is defined analogously to $V_{s,n}(S, W)$ where the admissible subset P^* is used in place of P, i.e.,

$$V_{s,n}^*(S, W) = (V_{s,n}^*(S) \setminus \bigcup_{w \in W} [w]) \cup (V_{s,n}^*(S) \cap W),$$

where $V_{s,n}^*(S) = \{u \in \Gamma^{[s]} \mid (\forall \langle q, a\rangle \in S)[\langle q, a, u\rangle \in P_n^*]\}$. Further, consider the sets $V = \{\langle 0^s, 0^n, S, W, u\rangle \mid u \in V_{s,n}(S, W)\}$ and $V^* = \{\langle 0^s, 0^n, S, W, u\rangle \mid u \in V_{s,n}^*(S, W)\}$ and note that since $C \in \mathrm{P}$, $P \in \Sigma_2^P$, and $P^* \in \mathrm{NP}$, it follows that V is in Σ_2^P and V^* is in NP. Since $\|\mathcal{V}_{s,n}(S)\| > 1$ implies that $V_{s,n}(S, W)$ 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 with 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 easily 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 computations we use the already mentioned approximate counting techniques, which we summarize in the following lemma (cf. (12)).

Lemma 20 *Let $f \in \min \sharp\mathrm{NP} \cup \max \sharp\mathrm{NP}$.*

(1) There exists an oracle $A \in \Sigma_2^P$ such that for all $x \in \Sigma^*$, and for all integers $d \geq 0$ and $e \geq 1$:
(a) $\langle x, d, 0^e \rangle \in A \Longrightarrow f(x) \leq (1 + 1/e)d$
(b) $\langle x, d, 0^e \rangle \notin A \Longrightarrow f(x) > d$.
(2) There exists a function $\tilde{f} \in \mathrm{FP}(\Sigma_2^P)$ such that for all $x \in \Sigma^*$, and for all integers $e \geq 1$, $f(x) \leq \tilde{f}(x, 0^e) \leq (1 + 1/e)f(x)$.

Finally, we show in the following lemma how to compute a good approximation $\tilde{\mu}$ to μ, a concept name u of sufficiently large weight when $\mu \geq \frac{7}{12}$, and a good query q for $V^*_{s,n}(S,W)$ when $\mu \leq \frac{3}{4}$.

Lemma 21 *Let C and P be as in Theorem 14, and suppose that C has polynomial general dimension under P via the polynomials p and m. Then, on input $\langle 0^s, 0^n, S, W \rangle$ it is possible to compute the following in polynomial time with an oracle in Σ_2^P:*

(1) an approximation $\tilde{\mu}$ of the bias μ of $V^*_{s,n}(S,W)$ satisfying $|\tilde{\mu} - \mu| \leq \frac{1}{12}$,
(2) a concept name u of weight $\mu(u) \geq \frac{1}{2}$ in $V^*_{s,n}(S,W)$, provided that $\mu \geq \frac{7}{12}$, and
(3) a query q of length $m(s,n)$ that is $1/6p(s,n)$-good for $V^*_{s,n}(S,W)$, provided that $\mu \leq \frac{3}{4}$.

PROOF. Let t be the cardinality of the set $V^*_{s,n}(S,W)$. Since t can be expressed as

$$t = \|\{v \in \Gamma^{[s]} \mid \langle 0^s, 0^n, S, W, v \rangle \in V^*\}\|,$$

where the set $V^* = \{\langle 0^s, 0^n, S, W, v \rangle \mid v \in V^*_{s,n}(S,W)\}$ is in NP, it follows that t considered as a function of $\langle 0^s, 0^n, S, W \rangle$ is in $\sharp$NP. Hence, Lemma 20 gives us an approximation $\tilde{t}$ computable in polynomial time with an oracle in Σ_2^P, such that $t \leq \tilde{t} \leq (1 + 1/36p(s,n))t$.

Now let $b = \min_{|u| \le s} b(u)$, where

$$\begin{aligned} b(u) &= \|V^*_{s,n}(S, W) \setminus [u]\| \\ &= \|\{v \mid \langle 0^s, 0^n, S, W, v\rangle \in V^* \wedge (\exists x \in \Sigma^n)[x \in \kappa_n(v) \triangle \kappa_n(u)]\}\|. \end{aligned}$$

Clearly, $b(u)$ considered as a function of $\langle 0^s, 0^n, S, W, u\rangle$ is in $\sharp$NP and hence, b considered as a function of $\langle 0^s, 0^n, S, W\rangle$ is in min$\,\sharp$NP. Thus, Lemma 20 gives us an approximation $\tilde{b}$ computable in polynomial time with an oracle in Σ_2^P such that $b \le \tilde{b} \le (1 + 1/36)b$.

Now the bias μ of $V^*_{s,n}(S, W)$ can be expressed as

$$\mu = \max_{|u| \le s} \frac{\|V^*_{s,n}(S, W) \cap [u]\|}{\|V^*_{s,n}(S, W)\|} = 1 - \min_{|u| \le s} \frac{\|V^*_{s,n}(S, W) \setminus [u]\|}{\|V^*_{s,n}(S, W)\|} = 1 - \frac{b}{t}.$$

Letting $\tilde{\mu} = 1 - \tilde{b}/\tilde{t}$, it follows that

$$\tilde{\mu} = \frac{\tilde{t} - \tilde{b}}{\tilde{t}} \le \frac{t + t/36 - b}{t} = 1 + \frac{1}{36} - \frac{b}{t} = \mu + \frac{1}{36},$$

and since $b \le t$ it also holds that

$$\tilde{\mu} = 1 - \frac{\tilde{b}}{\tilde{t}} \ge 1 - \frac{b + b/36}{t} = 1 - \frac{b}{t} - \frac{b}{36t} \ge \mu - \frac{1}{36} \ge \mu - \frac{1}{12}.$$

This shows (1).

Now let us see how to construct a concept name u of large weight in $V^*_{s,n}(S, W)$ if $\mu \ge \frac{7}{12}$. Since $b(u)$ as a function of $\langle 0^s, 0^n, S, W, u\rangle$ is computable in $\sharp$NP, Lemma 20 gives us an oracle A in Σ_2^P such that for all integers $d \ge 0$,

(1) $\langle 0^s, 0^n, S, W, u, d\rangle \in A \Longrightarrow b(u) \le (1 + \frac{1}{36})d$,
(2) $\langle 0^s, 0^n, S, W, u, d\rangle \notin A \Longrightarrow b(u) > d$.

Let $A' \in \Sigma_2^P$ be the oracle consisting of all tuples $\langle 0^s, 0^n, S, W, u', d\rangle$ for which there exists some $u \in \Gamma^{[s]}$ with prefix u' and $\langle 0^s, 0^n, S, W, u, d\rangle \in A$. Since there exists some concept name $u \in \Gamma^{[s]}$ with $b(u) \le b \le \tilde{b}$, and hence $\langle 0^s, 0^n, S, W, u, \tilde{b}\rangle \in A$, it follows that we can compute a concept name u with $\langle 0^s, 0^n, S, W, u, \tilde{b}\rangle \in A$ by prefix search in polynomial time with the help of oracle A'. Since for the resulting u it holds that $\langle 0^s, 0^n, S, W, u, \tilde{b}\rangle \in A$, it follows that

$$b(u) \le (1 + \frac{1}{36})\tilde{b} \le (1 + \frac{1}{36})^2 b \le (1 + \frac{1}{12})b$$

and since $b \le t$ and $\mu \ge \frac{7}{12}$, we get

$$\mu(u) = 1 - \frac{b(u)}{t} \ge 1 - \frac{b + b/12}{t} \ge \mu - \frac{1}{12} \ge \frac{1}{2}.$$

Finally, we have to show how to construct a good query q under the assumption $\mu \leq \frac{3}{4}$. For this consider the function

$$\begin{aligned} c(q) &= \max_{|a| \leq l(n,|q|)} \|\{u \in V^*_{s,n}(S,W) \mid \langle q,a,u\rangle \in P^*_n\}\| \\ &= \max_{|a| \leq l(n,|q|)} \|\{u \mid \langle 0^s, 0^n, S, W, u\rangle \in V^* \wedge \langle 0^n, q, a, u\rangle \in P^*\}\| \end{aligned}$$

Since $c(q)$ considered as a function of $\langle 0^s, 0^n, S, W, q\rangle$ is computable in $\max\sharp\mathrm{NP}$, Lemma 20 gives us an oracle B in Σ^P_2 such that for all integers $d \geq 0$,

(1) $\langle 0^s, 0^n, S, W, q, d\rangle \in B \Longrightarrow c(q) \leq (1 + \frac{1}{36p(s,n)})d$,
(2) $\langle 0^s, 0^n, S, W, q, d\rangle \notin B \Longrightarrow c(q) > d$.

By Lemma 19, the assumption $\mu \leq 3/4$ 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

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

and hence

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

Fixing d to be $d = (1 - 1/4p(s,n))\tilde{t}$, it follows that $\langle 0^s, 0^n, S, W, q, d\rangle \in B$, and similarly as in the proof of part 2 of the lemma we can construct a query q in polynomial time with the help of an oracle in Σ^P_2 such that $\langle 0^s, 0^n, S, W, q, d\rangle \in B$. Now, for the resulting query q, $\langle 0^s, 0^n, S, W, q, d\rangle \in B$ implies

$$\begin{aligned} c(q) &\leq (1 + \frac{1}{36p(s,n)})(1 - \frac{1}{4p(s,n)})\tilde{t} \\ &\leq (1 + \frac{1}{36p(s,n)})^2(1 - \frac{1}{4p(s,n)})t \\ &\leq (1 - \frac{1}{4p(s,n)} + \frac{1}{12p(s,n)})t \\ &= (1 - \frac{1}{6p(s,n)})t. \end{aligned}$$

Since $c(q)$ 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

$$\begin{aligned}
&\|\{u \in V^*_{s,n}(S,W) \mid \langle q,a,u\rangle \notin P_n\}\| \\
&\qquad = \|\{u \in V^*_{s,n}(S,W) \mid \langle q,a,u\rangle \notin P^*_n\}\| \\
&\qquad = \|V^*_{s,n}(S,W)\| - \|\{u \in V^*_{s,n}(S,W) \mid \langle q,a,u\rangle \in P^*_n\}\| \\
&\qquad \geq \|V^*_{s,n}(S,W)\| - (1 - \frac{1}{6p(s,n)})\|V^*_{s,n}(S,W)\| \\
&\qquad = \frac{1}{6p(s,n)}\|V^*_{s,n}(S,W)\|.
\end{aligned}$$

However, the assumption $\mu \leq 3/4$ does not only provide a good query but also implies that for all answers a with $\|\Lambda_n(q,a)\| = 1$ it holds that

$$\begin{aligned}
\|\{u \in V^*_{s,n}(S,W) \mid \langle q,a,u\rangle \in P_n\}\| &\leq \max_{|u|\leq s}\|[u] \cap V^*_{s,n}(S,W)\| \\
&\leq (\frac{3}{4})\|V^*_{s,n}(S,W)\|
\end{aligned}$$

and hence

$$\|\{u \in V^*_{s,n}(S,W) \mid \langle q,a,u\rangle \notin P_n\}\| \geq (\frac{1}{4})\|V^*_{s,n}(S,W)\|$$

This concludes the proof of the lemma. □

5 Non-learnability with an oracle in NP

In this section we show that the Σ_2^P oracle in our main theorem cannot be replaced by an oracle in NP, unless the polynomial hierarchy collapses to Δ_2^P. This computational hardness result is based on the *representation problem* REP(C) (13; 8) 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 (cf. Example 2).

Aizenstein et al. (8) showed that there is a representation $K \in$ P such that its representation problem REP(K) is complete for Σ_2^P. The representation K can be described as follows. Concept names are boolean circuits of the form $c \vee t$, where c is an arbitrary 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 $x \in \{0,1\}^n$ accepted by $c \vee t$. (In (8) a 3CNF formula is used in place of a circuit c 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 $\mathrm{Eq}(K)$ needs at least $2^{n/2} - 1$ queries to identify the target.

Proposition 22 *K does not have polynomial learning complexity under* $\mathrm{Eq}(K)$.

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 23 *K has polynomial learning complexity under* $\mathrm{Eq}(\mathrm{Circ})$ *and hence, K is polynomial-time learnable under* $\mathrm{Eq}(Circ)$ *with an oracle in* Σ_2^P.

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 prove the following theorem.

Theorem 24 *K is not polynomial-time learnable under* $\mathrm{Eq}(Circ)$ *with an oracle in* NP*, unless the polynomial-time hierarchy collapses to* Δ_2^P.

PROOF. If there is a learning algorithm A for K under $\mathrm{Eq}(Circ)$ whose running time is bounded by $p(n, s)$ for some polynomial p and which uses an oracle in NP, then we can solve the representation problem for K in Δ_2^P as follows. On input $\langle 0^s, 0^n, c\rangle$ run A on input $\langle 0^s, 0^n\rangle$ and answer each equivalence query $h \vee t$ of A by some counterexample $x \in \{0,1\}^n$ with $c(x) \neq h(x) \vee t(x)$, if it exists. Clearly, x can be found in polynomial time with the help of an NP oracle. Then $\langle 0^s, 0^n, c\rangle$ belongs to $\mathrm{REP}(K)$ if and only if A succeeds within $p(n, s)$ steps, implying that the representation problem for K is in Δ_2^P. Combining this with the Σ_2^P-completeness of $\mathrm{REP}(K)$ we get the desired collapse of the polynomial-time hierarchy. $\square$

Thus we have found representations C and P satisfying the conditions of Theorem 14 but C is not polynomial-time learnable under P with an oracle in NP, unless the polynomial hierarchy collapses to Δ_2^P. In fact, by Theorem 14, 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 hypotheses are of the form $c \vee t$, we now consider learning with *projective* equivalence queries (14; 15; 7; 16). 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.

It is easy to see that Proj-Eq(K) is decidable in coNP. Hence, Corollary 15 implies that K is polynomial-time learnable under Proj-Eq(K) with an oracle in Σ_3^P. However, by exploiting special properties of the concept class K we can show the following theorem.

Theorem 25 *K is polynomial-time learnable under* Proj-Eq(K) *with an oracle in Σ_2^P.*

PROOF. We only roughly sketch the learning algorithm A. On input $\langle 0^s, 0^n \rangle$, A computes two circuits c_0 and c_1 such that c_0 agrees with the target $c \vee t$ on all $x = x_1 \cdots x_n$ with $x_1 = 0$ and c_1 agrees with the target on all x with $x_1 = 1$. By using a Σ_2^P oracle, A can then determine a hypothesis that is equivalent to the target. It remains to argue that the circuits c_b, $b \in \{0, 1\}$, can be found in polynomial-time under Proj-Eq(K) with an oracle in Σ_2^P.

Clearly, by using a simple majority-vote strategy it is possible to compute c_b in polynomial-time with an oracle in Σ_2^P by asking projective equivalence queries of the form $\langle b* \cdots *, h \rangle$, where h is an arbitrary circuit. Thus it suffices to observe that any projective equivalence query $\langle b* \cdots *, h \rangle$ can be simulated by the query $\langle b * \cdots *, h \vee t_b \rangle$, where $t_0 = x_1 \wedge x_2 \wedge \cdots \wedge x_{n/2}$ and $t_1 = \overline{x}_1 \wedge x_2 \wedge \cdots \wedge x_{n/2}$. $\square$

Similarly as in Theorem 24, the learnability of K under Proj-Eq(K) with an oracle in NP implies that the representation problem of K is in Δ_2^P. Thus, we also have the following analogue of Theorem 24.

Theorem 26 *K is not polynomial-time learnable under* Proj-Eq(K) *with 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] D. Angluin, Queries and concept learning, Machine Learning 2 (1988) 319–342.

[2] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, C. Tamon, Oracles and queries that are sufficient for exact learning, J. Comput. Syst. Sci. 52 (1996) 421–433.

[3] L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan, D. Wilkins, How many queries are needed to learn?, J. ACM 43 (5) (1996) 840–862.

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

[5] M. J. Kearns, L. G. Valiant, Cryptographic limitations on learning boolean formulae and finite automata, J. ACM 41 (1994) 67–95.

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

[7] J. L. Balcázar, J. Castro, D. Guijarro, A new abstract combinatorial dimension for exact learning via queries, Journal of Computer and System Sciences 64 (2002) 2–21.

[8] H. Aizenstein, T. Hegedüs, L. Hellerstein, L. Pitt, Complexity theoretic hardness results for query learning, Computational Complexity 7 (1998) 19–53.

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

[10] O. Watanabe, A framework for polynomial time query learnability, Mathematical Systems Theory 27 (1994) 211–229.

[11] D. Angluin, M. Kharitonov, When won't membership queries help?, J. Comput. Syst. Sci. 50 (1995) 336–355.

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

[13] O. Watanabe, R. Gavaldà, Structural analysis of polynomial time query learnability, Mathematical Systems Theory 27 (1994) 231–256.

[14] L. Hellerstein, M. Karpinsky, Learning read-once formulas using membership queries, in: Proc. 2nd ACM Conference on Computational Learning Theory, Morgan Kaufmann, 1989, pp. 146–161.

[15] W. Maass, G. Turan, On the complexity of learning from counterexam-

ples, in: Proc. 30th IEEE Symposium on the Foundations of Computer Science, IEEE Computer Society Press, 1989, pp. 262–267.

[16] D. Angluin, L. Hellerstein, M. Karpinski, Learning read-once formulas with queries, Journal of the ACM 40 (1) (1993) 185–210.