Parameterized Learnability of k-Juntas and Related Problems*

Vikraman Arvind[1], Johannes Köbler[2], and Wolfgang Lindner[3]

[1] The Institute of Mathematical Sciences, Chennai 600 113, India
arvind@imsc.res.in
[2] Institut für Informatik, Humboldt Universität zu Berlin, Germany
koebler@informatik.hu-berlin.de
[3] Sidonia Systems, Grubmühl 20, D-82131 Stockdorf, Germany
wolfgang.lindner@sidoniasystems.de

Abstract. We study the parameterized complexity of learning k-juntas and some variations of juntas. We show the hardness of learning k-juntas and subclasses of k-juntas in the PAC model by reductions from a W[2]-complete problem. On the other hand, as a consequence of a more general result we show that k-juntas are exactly learnable with improper equivalence queries and access to a W[P] oracle.

Subject Classification: Learning theory, computational complexity.

1 Introduction

Efficient machine learning in the presence of irrelevant information is an important issue in computational learning theory (see, e.g., [21]). This has motivated the fundamental problem of learning *k-juntas*: let f be an unknown boolean function defined on the domain $\{0,1\}^n$ that depends only on an unknown subset of at most k variables, where $k \ll n$. Such a boolean function f is referred to as a *k-junta*, and the problem is whether this class of functions is efficiently learnable (under different notions of learning). This is a natural parameterized learning problem that calls for techniques from parameterized complexity.

Our study is motivated by the recent exciting work by Mossel, O'Donnell and Servedio [22] and the article with open problems on k-juntas proposed by Blum [4,3], drawing to our attention the connection between the learnability of k-juntas and fixed parameter tractability. Notice that in the distribution-free PAC model, an exhaustive search algorithm can learn k-juntas in time roughly n^k. For the uniform distribution, [22] have designed an algorithm for learning k-juntas in time roughly $n^{0.7 \cdot k}$. For the smaller class of monotone k-juntas they even achieve a running time polynomial in n and 2^k (for this class an algorithm with a different running time is given in [8]). Further, for learning symmetric k-juntas, Lipton et al. [20] have provided an algorithm with running-time roughly $n^{0.1 \cdot k}$ and this bound has been subsequently improved to $O(n^{k/\log k})$ in [18].

* Work supported by a DST-DAAD project grant for exchange visits.

M. Hutter, R.A. Servedio, and E. Takimoto (Eds.): ALT 2007, LNAI 4754, pp. 120–134, 2007.

Actually, natural parameters abound in the context of learning and several other learning algorithms in the literature can be seen as parameterized learning algorithms. We mention only two important further examples: Kushilevitz and Mansour [19, Theorem 5.3] give an exact learning algorithm with membership queries for boolean decision trees of depth d and n variables with $\mathbb{F}_2$-linear functions at each node with running time polynomial in n and 2^d. Blum and Rudich [5] design an exact learning algorithm with (improper) equivalence and membership queries for k-term DNFs which runs in time $n2^{O(k)}$.

Parameterized Complexity, introduced as an approach to coping with intractability by Downey and Fellows in [11], is now a flourishing area of research (see, e.g. the monographs [12,14]). Questions focussing on parameterized problems in computational learning have been first studied in [10]. Fixed parameter tractability provides a notion of feasible computation less restrictive than polynomial time. It provides a theoretical basis for the design of new algorithms that are efficient and practically useful for small parameter values. We quickly recall the rudiments of this theory relevant for the present paper. More details (especially on the levels of the W-hierarchy) will be given in the next section (see also [12,14]).

Computational problems often have inputs consisting of two or more parts where some of these parts typically take only small values. For example, an input instance of the vertex cover problem is (G, k), and the task is to determine if the graph G has a vertex cover of size k. A similar example is the k-clique problem where again an input instance is a pair (G, k) and the problem is to test if the graph G has a clique of size k. For such problems an exhaustive search will take time $O(n^k)$, where n is the number of vertices in G. However, a finer classification is possible. The vertex cover problem has an $2^k n^{O(1)}$ time algorithm, whereas no algorithm is known for the k-clique problem of running time $O(n^{o(k)})$. Thus, if the parameter k is such that $k \ll n$, then we have a faster algorithm for the k-vertex cover problem than is known for the k-clique problem.

More generally, a *parameterized decision problem* is a pair (L, κ) where $L \subseteq \{0,1\}^*$ and κ is a polynomial time computable function $\kappa : \{0,1\}^* \to \mathbb{N}$. We call $k = \kappa(x)$ the parameter value of the instance x. The problem (L, κ) is *fixed parameter tractable* ($(L, \kappa) \in$ FPT for short) if L is decidable by an *fpt algorithm*, i.e., by an algorithm that runs in time $g(\kappa(x))|x|^{O(1)}$ for an arbitrary computable function g. In particular, the k-vertex cover problem has an $2^k n^{O(1)}$ time algorithm, implying that it is fixed parameter tractable. On the other hand, the k-clique problem is not known to be in FPT.

In their seminal work, Downey and Fellows [11,12] also developed a theory of intractability for parameterized problems as a tool to classify parameterized problems according to their computational hardness. The W-hierarchy consists of the levels $W[t]$, $t \geq 1$, together with the two classes W[SAT] and W[P] and we have the inclusions

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \cdots \subseteq \text{W}[\text{SAT}] \subseteq \text{W}[\text{P}].$$

In this paper, we show that k-juntas and some subclasses of k-juntas are proper PAC learnable in fixed parameter time with access to an oracle in the

second level W[2] of the W-hierarchy. This bound is achieved by reducing the parameterized consistency problem for k-juntas to the parameterized set cover problem. In order to achieve proper learning in fixed parameter time, the learner computes an optimal set cover with the help of a W[2] oracle. A similar approach has been used by Haussler [15] to design an efficient PAC-learning algorithm for k-monomials using $O(\varepsilon^{-1}(\log(\delta^{-1}) + k\log(n)(\log(k) + \log\log(n))))$ many examples.

As a lower bound we prove that monotone k-monomials are not even PAC learnable with k-juntas as hypotheses in randomized fixed parameter time unless W[2] has randomized FPT algorithms. The proof is an application of the well-known technique introduced by Pitt and Valiant [23] to reduce a hard problem to the consistency problem for the hypothesis class. Further, we describe a deterministic fpt algorithm that proper PAC learns k-monomials under the uniform distribution.

We next consider the question of exactly learning k-juntas with only equivalence queries. It turns out that k-juntas are learnable by a randomized fpt algorithm with improper equivalence queries and access to a W[P] oracle. As a consequence, k-juntas are also fpt PAC learnable with access to a W[P] oracle. Actually, we prove a more general result: we consider the problem of learning parameterized concept classes for which the membership of an assignment to a given concept is decidable in FPT and show that these concept classes are exactly learnable by a randomized fpt algorithm with equivalence queries and with access to a W[P] oracle, provided that the Hamming weight is used as parameter. Our learning algorithm uses a similar strategy as the algorithm designed by Bshouty et al. [7] for exactly learning boolean circuits with equivalence queries and with the help of an NP oracle.

The rest of the paper is organized as follows. In Section 2 we provide the necessary notions and concepts and fix notation. Section 3 contains our results on PAC learning and in Section 4 we prove the query-learning results.

2 Preliminaries

2.1 Parameterized Complexity

We fix the alphabet $\Sigma = \{0, 1\}$. The *Hamming weight* $w(x)$ of a string $x \in \{0, 1\}^*$ is the number of 1's in x. The cardinality of a finite set X is denoted by $\|X\|$.

The key idea in quantifying parameterized hardness is the notion of the *weft* of a boolean circuit [11]: We fix any constant $l > 2$. In a boolean circuit c we say that a gate is *large* if it has fanin at least l. The *weft* of a boolean circuit (or formula) c is the maximum number of large gates on any input to output path in c. Thus, any CNF formula is a depth 2 and weft 2 circuit, whereas k-CNF formulas (i.e. CNF formulas with at most k literals per clause) are circuits of depth 2 and weft 1.

The following parameterized problem Weighted-Circuit-SAT (a weighted version of the satisfiability problem for boolean circuits) is central to this theory: Given a pair (c, k), where c is a boolean circuit (or formula) and $k =$

$\kappa(c,k)$ is the parameter, the problem is to decide if there is an input of hamming weight k accepted by c. For a class C of circuits we denote the parameterized problem Weighted-Circuit-SAT restricted to circuits from C by Weighted-Circuit-SAT(C).

In order to compare the complexity of parameterized problems we use the fpt many-one and Turing reducibilities [11]. An *fpt many-one reduction* f from a parameterized problem (L,κ) to a parameterized problem (L',κ') maps an instance x for L to an equivalent instance $f(x)$ for L' (i.e., $x \in L \Leftrightarrow f(x) \in L'$), where for a computable function g, $f(x)$ can be computed in time $g(\kappa(x))|x|^{O(1)}$ and $\kappa'(f(x))$ is bounded by $g(\kappa(x))$. The notion of an *fpt Turing reduction* where the parameterized problem (L',κ') is used as an oracle is defined accordingly: An *fpt Turing reduction* from a parameterized problem (L,κ) to a parameterized problem (L',κ') is a deterministic algorithm $\mathcal{M}$ that for a computable function g, decides L with the help of oracle L' in time $g(\kappa(x))|x|^{O(1)}$ and asks only queries y with $\kappa'(y) \leq g(\kappa(x))$. Now we are ready to define the *weft hierarchy* and the class XP [12,14].

- For each $t \geq 1$, W[t] is the class of parameterized problems that for some constant d are fpt many-one reducible to the weighted satisfiability problem for boolean formulas of depth d and weft t.
- The class W[SAT] consists of parameterized problems that are fpt many-one reducible to the weighted satisfiability problem for boolean formulas.
- W[P] is the class of parameterized problems fpt many-one reducible to the weighted satisfiability problem for boolean circuits.
- For each $k \in \mathbb{N}$, the k^{th} slice of a parameterized problem (L,κ) is the language $L_k = \{x \in L \mid \kappa(x) = k\}$. A parameterized problem (L,κ) belongs to the class XP if for any k, the k^{th} slice L_k of (L,κ) is in P. Note that XP is a non-uniform class that even contains undecidable problems. There is also a uniform version of XP that is more suitable for our purpose. A parameterized problem (L,κ) belongs to the class uniform-XP if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm that, given $x \in \{0,1\}^*$, decides if $x \in L$ in at most $|x|^{f(\kappa(x))} + f(\kappa(x))$ steps.

From these definitions it is easy to see that we have the following inclusion chain:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \cdots \subseteq \text{W[SAT]} \subseteq \text{W[P]} \subseteq \text{uniform-XP} \subseteq \text{XP}.$$

2.2 Parameterized Learnability

The Boolean constants *false* and *true* are identified with 0 and 1, and B_n denotes the set of all Boolean functions $f : \{0,1\}^n \to \{0,1\}$. Elements x of $\{0,1\}^n$ are called *assignments* and any pair (x,b) with $f(x) = b$ is called an *example of* f. A variable x_i is called *relevant* for f, if there is an assignment x with $f(x) \neq f(x')$, where x' is obtained from x by flipping the i-th bit.

In order to make our presentation concise, we only consider learning of *concept classes* $C \subseteq B_n$ for some fixed arity n. By abusing notation, we often identify a concept $f \in C$ with the set $\{x \in \{0,1\}^n \mid f(x) = 1\}$.

A *representation of concepts* is a set $R \subseteq \{0,1\}^*$ of encoded pairs $\langle r, x\rangle$. A *concept name* r represents for each integer $n \geq 1$ the concept

$$R_n(r) = \{x \in \{0,1\}^n \mid \langle r, x\rangle \in R\}.$$

The concept class represented by R is $C(R) = \bigcup_{n\geq 1} C_n(R)$ where $C_n(R) = \{R_n(r) \mid r \in \{0,1\}^*\}$.

A *parameterization* of a representation R of concepts is a polynomial-time computable function $\kappa : \{0,1\}^* \to \mathbb{N}$. We call (R, κ) a *parameterized representation* of concepts and $k = \kappa(r)$ the *parameter value* of the concept description r. (R, κ) is said to be *fpt evaluable* if $\langle r, x\rangle \in R$ is decidable in time $g(\kappa(r))p(|r|, |x|)$, for some arbitrary computable function g and some polynomial p. For a pair of integers k, s we denote by $R_{k,s}$ the set $\{r \in \{0,1\}^s \mid \kappa(r) = k\}$ of all representations r of size s having parameter value k.

The concept classes we consider in the present paper are the following.

- The class $\cup_{n>0} B_n$ of all boolean functions. We usually represent these concepts by (binary encodings of) boolean circuits.
- The class $J_{k,n}$ of *k-juntas* in B_n. If we represent k-juntas by boolean circuits c, then we use the number k of input gates x_i in c having fanout at least 1 as parameter. As in [1] we can also represent concepts in $J_{k,n}$ by strings of length $n + 2^k$ having at most $k + 2^k$ ones, where the first part is of length n and contains exactly k ones (specifying the relevant variables) and the second part consists of the full value table of the k-junta. We denote this representation of k-juntas by J.
- Likewise, for the class $M_{k,n}$ of *k-monomials* consisting of all conjunctions f of at most k literals, we can represent f by a string of length $n+k$ having at most $2k$ ones, where the first n bits specify the set of relevant variables of f (exactly as for k-juntas) and the last k bits indicate which of these variables occur negated in f. Clearly, monotone k-juntas $f \in \mathit{mon}\text{-}J_{k,n}$ and monotone k-monomials $f \in \mathit{mon}\text{-}M_{k,n}$ can be represented in a similar way.

The Hamming weight $w(r)$ provides a natural parameterization of concept classes $R_n(r)$. In fact, if we use the representation J of k-juntas described above, then this parameterization is equivalent to the usual one since for every string r representing a k-junta it holds that $k \leq w(r) \leq k + 2^k$. Further, it is easy to see that all parameterized representations considered in this paper are fpt (even polynomial-time) evaluable. W.r.t. the Hamming weight parameterization, notice that $R_{k,s}$ has size $s^{O(k)}$. Furthermore, the set $R_{k,s}$ can be easily enumerated in time $s^{O(k)}$. This motivates the following definition: a parameterized representation (R, κ) is XP-*enumerable* if the set $R_{k,s}$ can be enumerated in time $s^{O(k)}$ by a uniform algorithm.

Valiant's model of probably approximately correct (PAC) learning [25] and Angluin's model of exact learning via queries [2] are two of the most well-studied models in computational learning theory. In the parameterized setting, both PAC-learning and exact learning with queries are defined in the standard way. However, the presence of the fixed parameter allows a finer complexity classification of learning problems.

To define a parameterized version of exact learning with equivalence queries, let (R, κ) and H be (parameterized) representations. An algorithm $\mathcal{A}$ *exactly learns* (R, κ) *using equivalence queries from* H, if for all $n \in \mathcal{N}$ and all concept names r,

1) $\mathcal{A}$ gets inputs n, $s = |r|$ and $k = \kappa(r)$.
2) $\mathcal{A}$ makes equivalence queries with respect to $R_n(r)$, where the query is a concept name $h \in \{0,1\}^*$, and the answer is either "Yes" if $H_n(h) = R_n(r)$ or a counterexample x in the symmetric difference $H_n(h) \triangle R_n(r)$.
3) $\mathcal{A}$ outputs a concept name $h \in \{0,1\}^*$ such that $H_n(h) = R_n(r)$.

We say that $\mathcal{A}$ is an *fpt EQ-learning algorithm* if for each integer $n \in \mathcal{N}$ and each target r the running time of $\mathcal{A}$ on input n, $s = |r|$ and $k = \kappa(r)$ is bounded by $g(k)p(n,s)$, for some computable function g and some polynomial p.

Next we define parameterized PAC-learning. Let (R, κ) and H be (parameterized) representations. A (possibly randomized) algorithm $\mathcal{A}$ *PAC-learns* (R, κ) *using hypotheses from* H, if for all $n \in \mathcal{N}$, all concept names r and for all $\epsilon, \delta > 0$,

1) $\mathcal{A}$ gets inputs n, $s = |r|$, $k = \kappa(r)$, ϵ and δ.
2) $\mathcal{A}$ gets random examples (x, b) of the concept $R_n(r)$, where the strings x are chosen independently according to some distribution $\mathcal{D}_n$ on $\{0,1\}^n$.
3) With probability at least $1 - \delta$, $\mathcal{A}$ outputs a concept name $h \in \{0,1\}^*$ such that the error
$$error(h) = \Pr_{x \in \mathcal{D}_n}[x \in R_n(r) \triangle H_n(h)]$$
of h with respect to the target r, where x is chosen according to $\mathcal{D}_n$, is at most ϵ.

$\mathcal{A}$ is an *fpt algorithm* if for each integer $n \in \mathcal{N}$, each target r and for all $\epsilon, \delta > 0$, the running time of $\mathcal{A}$ is bounded by $g(k)p(n, s, 1/\epsilon, 1/\delta)$, for an arbitrary computable function g and a polynomial p. We say that (R, κ) is *fpt PAC-learnable with hypotheses from* H, if there is an fpt algorithm $\mathcal{A}$ that PAC-learns (R, κ) using hypotheses from H.

As usual, in *distribution-free* PAC-learning, the algorithm must succeed on any unknown distribution, whereas in *distribution-specific* PAC-learning the learning algorithm only works for a fixed distribution.

3 PAC Learning of k-Juntas

By the classical algorithm due to Haussler [15] (using the modification of Warmuth as described in [17, Chapter 2]), the class of k-monomials is PAC-learnable in time $poly(n, 1/\varepsilon, \log(1/\delta))$ with $k\log(2/\varepsilon)$-monomials as hypotheses and using $O(\varepsilon^{-1}(\log(\delta^{-1}) + k\log(n)\log(\varepsilon^{-1})))$ many examples. The algorithm uses the well-known greedy heuristic to approximate the set cover problem [16,9]. By computing an optimal solution of the set cover problem, we can achieve proper learning with k-monomials as hypotheses, though at the expense of access to a

W[2] oracle. In the fixed parameterized setting, this can be extended to the class of k-juntas as well as to monotone k-monomials and to monotone k-juntas.

We first show that the parameterized consistency problem (see Definition 1) for (monotone) k-juntas and for (monotone) k-monomials is in W[2]. For this we use the parameterized version of the set cover problem defined as follows. Given a set $U = \{u_1, \ldots, u_m\}$, a family $S = \{S_1, \ldots, S_n\}$ of subsets $S_i \subseteq U$, and a positive integer k (which is the parameter), is there a subset $R \subseteq S$ of size k whose union is U. It is well-known that this problem is W[2]-complete (see for example the book [12]).

Definition 1. *The parameterized consistency problem for a concept class* $\mathcal{C} = \bigcup_{n\geq 1} C_n$*, where* $C_n \subseteq B_n$*, is defined as follows. Given sets* P *and* N *of positive and negative examples from* $\{0,1\}^n$ *and a positive integer* k *(which is the parameter), does* C_n *contain a* k*-junta* f *which is* consistent *with* P *and* N *(meaning that* $f(x) = 1$ *for all* $x \in P$ *and* $f(x) = 0$ *for all* $x \in N$*).*

Theorem 2. *The parameterized consistency problem is in* W[2] *for the following concept classes* $\mathcal{C} = \bigcup_{n\geq 1} C_n$*:*

1) *for all* k*-juntas (i.e.,* $C_n = \bigcup_{k=0}^{n} J_{k,n} = B_n$*),*
2) *for monotone* k*-juntas (i.e.,* $C_n = \bigcup_{k=0}^{n} mon\text{-}J_{k,n}$*),*
3) *for* k*-monomials (i.e.,* $C_n = \bigcup_{k=0}^{n} M_{k,n}$*),*
4) *for monotone* k*-monomials (i.e.,* $C_n = \bigcup_{k=0}^{n} mon\text{-}M_{k,n}$*).*

Moreover, in each case, a representation for a consistent k*-junta* $f \in C_n$ *can be constructed (if it exists) in fixed parameter time relative to a* W[2] *oracle.*

Proof. 1) Let (P, N, k) be an instance of the consistency problem for k-juntas. We claim that there is a k-junta consistent with P and N if and only if there is an index set $I \subseteq [n]$ of size k such that

$$\forall (a,b) \in P \times N \; \exists i \in I : a_i \neq b_i. \tag{1}$$

The forward implication is immediate. For the backward implication let I be a size k index set fulfilling property (1) and consider the k-junta f defined by

$$f(x) = \begin{cases} 1, & \text{there exists an } a \in P \text{ s.t. for all indices } i \in I : x_i = a_i, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Then it is clear that $f(a) = 1$ for all $a \in P$. Further, since by property (1) no assignment $b \in N$ can agree with any $a \in P$ on I, it follows that f is also consistent with N. Thus we have shown that (P, N, k) is a positive instance of the consistency problem for k-juntas if and only if the weft 2 formula $\bigwedge_{(a,b)\in P\times N} \bigvee_{a_i \neq b_i} x_i$ has a satisfying assignment of weight k, implying that the consistency problem for k-juntas is in W[2].

In order to construct a consistent k-junta with the help of a W[2] oracle in time $poly(2^k, m, n)$, where $m = \|P \cup N\|$, note that there is also an easy reduction of the parameterized consistency problem to the parameterized set cover problem.

In fact, for each $i \in [n]$ consider the subset $S_i = \{(a,b) \in P \times N \mid a_i \neq b_i\}$ of $U = P \times N$. Then an index set $I \subseteq [n]$ fulfills property (1) if and only if the subfamily $R = \{S_i \mid i \in I\}$ covers U. Now observe that a set S_i is contained in a size k subfamily $R \subseteq \{S_1, \ldots, S_n\}$ covering U if and only if the set $U' = U \setminus S_i$ is covered by some size $k-1$ subfamily of $R' = \{S_1 \setminus S_i, \ldots, S_n \setminus S_i\}$. Thus, we can successively construct a cover R of size k (if it exists) by using kn oracle calls to the parameterized set cover problem. From R we immediately get an index set $I \subseteq [n]$ fulfilling property (1) and thus, a representation of the consistent k-junta f defined by Equation (2) can be computed in fixed parameter time relative to the parameterized set cover problem.

2) Similarly as above it follows that there is a monotone k-junta consistent with P and N if and only if there is an index set $I \subseteq [n]$ of size k fulfilling the property

$$\forall (a,b) \in P \times N \; \exists i \in I : a_i > b_i. \tag{3}$$

In this case, the monotone k-junta f derived from a size k index set I with property (3) has the form

$$f(x) = \begin{cases} 1, & \text{there exists an } a \in P \text{ s.t. for all indices } i \in I : x_i \geq a_i, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Thus, there is some monotone k-junta which is consistent with P and N if and only if the weft 2 formula $\bigwedge_{(a,b)\in P\times N} \bigvee_{a_i > b_i} x_i$ has a satisfying assignment of weight k, implying that also the consistency problem for monotone k-juntas is in W[2]. Further, a consistent monotone k-junta can be constructed by computing a size k solution for the set cover instance $(U, \{S_1, \ldots, S_n\})$, where $U = P \times N$ and $S_i = \{(a,b) \in U \mid a_i > b_i\}$ for $i = 1, \ldots, n$.

3) First observe that a monomial can only be consistent with a set P of positive assignments if it does not depend on any variable x_i such that P contains two examples a and a' with $a_i \neq a'_i$. Let $J = \{i \in [n] \mid \forall a, a' \in P : a_i = a'_i\}$ and let a be an arbitrary but fixed positive example from P. Then there is some k-monomial which is consistent with P and N if and only if there is an index set $I \subseteq J$ of size k fulfilling the property

$$\forall b \in N \; \exists i \in I : a_i \neq b_i. \tag{5}$$

Indeed, if $I \subseteq J$ has property (5), then the monomial $\bigwedge_{i\in I, a_i=1} x_i \wedge \bigwedge_{i\in I, a_i=0} \bar{x}_i$ is consistent with P and N. Thus, some k-monomial is consistent with P and N if and only if the weft 2 formula $\bigwedge_{b\in N} \bigvee_{i\in J, a_i\neq b_i} x_i$ has a satisfying assignment of weight k, implying that also the consistency problem for k-monomials is in W[2]. Further, a consistent k-monomial can be constructed by computing a size k solution for the set cover instance $(N, \{S_i \mid i \in J\})$, where $S_i = \{b \in N \mid a_i \neq b_i\}$ for $i = 1, \ldots, n$.

4) The reduction is very similar to the previous one. Observe that a monotone monomial can only be consistent with a set P of positive assignments if it does not depend on any variable x_i such that P contains an example a with $a_i = 0$. Let $J = \{i \in [n] \mid \forall a \in P : a_i = 1\}$. Then there is some monotone k-monomial

which is consistent with P and N if and only if there is an index set $I \subseteq J$ of size k fulfilling the property

$$\forall b \in N \; \exists i \in I : b_i = 0. \tag{6}$$

Indeed, if $I \subseteq J$ fulfills this property, then the monomial $\bigwedge_{i \in I} x_i$ is consistent with P and N. Thus, some k-monomial is consistent with P and N if and only if the weft 2 formula $\bigwedge_{b \in N} \bigvee_{i \in J} x_i$ has a satisfying assignment of weight k, implying that also the consistency problem for monotone k-monomials is in W[2]. Further, a consistent monotone k-monomial can be constructed by computing a size k solution for the set cover instance $(N, \{S_i \mid i \in J\})$, where $S_i = \{b \in N \mid b_i = 0\}$ for $i = 1, \ldots, n$. □

Theorem 3. *The class of k-juntas is fpt PAC-learnable with access to a* W[2] *oracle and using k-juntas as hypotheses. The same holds for monotone k-juntas as well as for k-monomials and monotone k-monomials.*

Proof. We first consider the case of k-juntas and monotone k-juntas. As has been observed in [1], the set of all k-juntas has size $O(n^k 2^{2^k})$ and hence it follows from [6] that (monotone) k-juntas are proper PAC-learnable by an Occam algorithm by using $O(\varepsilon^{-1}(\log(\delta^{-1}) + 2^k + k \log(n)))$ many examples. Further, observe that using the algorithm described in the proof of Theorem 2, a (monotone) k-junta consistent with the random training sample (P, N) can be constructed with the help of a W[2] oracle in time $poly(2^k, m, n)$, where $m = \|P \cup N\|$.

For the case of (monotone) k-monomials we note that the variant of Haussler's algorithm that requests $O(\varepsilon^{-1}(\log(\delta^{-1}) + k \log(n)))$ many examples and uses the parameterized set cover problem as an oracle to determine a consistent (monotone) k-monomial learns this class in time $poly(n, 1/\varepsilon, \log(1/\delta))$. □

In order to show that the W[2] oracle is indeed necessary we make use of the following hardness result that easily follows by transforming Haussler's [15] reduction of the set cover problem to the consistency problem for monotone monomials into the parameterized setting.

Lemma 4. *Let $\mathcal{C} = \bigcup_{n \geq 1} C_n$ be a concept class where C_n contains all monotone monomials over the variables $x_1, \ldots, x_n$. Then the parameterized consistency problem for $\mathcal{C}$ is hard for* W[2].

Proof. Consider Haussler's [15] reduction f that maps a set cover instance $U = \{u_1, \ldots, u_m\}$, $S = \{S_1, \ldots, S_n\}$ and k to the instance $P = \{1^n\}$, $N = \{b_1, \ldots, b_m\}$ and k, where the i-th bit of the negative example b_j is 0 if and only if $u_j \in S_i$. We claim that the following statements are equivalent:

- some k-junta is consistent with P and N,
- U can be covered by a subfamily $R \subseteq S$ of size k,
- some monotone k-monomial is consistent with P and N.

Suppose that some k-junta $f \in C_n$ is consistent with the examples from P and N. Let I be the index set of the relevant variables of f. Then by the choice

of $P = \{1^n\}$, each negative example b_j differs from 1^n in at least one of the k positions from I. This means that for every $j \in [m]$ there is some $i \in I$ such that the i-th bit of b_j is 0 and, hence, $u_j \in S_i$. Thus, the union of all sets S_i with $i \in I$ covers U.

Now suppose that U can be covered by a subfamily $R = \{S_i \mid i \in I\}$ for some index set $I \subseteq [n]$ of size k. Then for every $j \in [m]$ there is some index $i \in I$ such that the i-th bit of b_j is 0, implying that the monotone k-monomial $\bigwedge_{i \in I} x_i$ is false on all b_j from N and true on 1^n.

Since, by assumption, C_n contains all monotone monomials over the variables $x_1, \ldots, x_n$, this shows that f is an fpt many-one reduction of the parameterized set cover problem (which is W[2]-complete) to the parameterized consistency problem for $\mathcal{C}$. □

By combining Lemma 4 with Theorem 2 we immediately get the following completeness results.

Corollary 5. *The parameterized consistency problem for the following concept classes is complete for* W[2]*:*

1) *all k-juntas,*
2) *monotone k-juntas,*
3) *k-monomials,*
4) *monotone k-monomials.*

Next we show that no concept class containing all monotone k-monomials is fpt PAC-learnable with boolean circuits having at most k relevant variables as hypotheses unless the second level of the W-hierarchy collapses to randomized FPT (meaning that for any problem $(L, \kappa) \in \mathrm{W}[2]$ there is a randomized algorithm that decides L in expected time $g(\kappa(x))|x|^{O(1)}$ for a computable function g; see [13]).

Theorem 6. *Monotone k-monomials are not fpt PAC-learnable with boolean circuits having at most k relevant variables as hypotheses, unless* W[2] *is contained in randomized* FPT.

Proof. Assume that there exists a PAC-learning algorithm $\mathcal{A}$ for the set of monotone k-monomials which runs in time $g(k)poly(n, 1/\varepsilon, 1/\delta)$ and outputs boolean circuits with at most k relevant variables as hypotheses. We describe a randomized algorithm $\mathcal{M}$ which solves the parameterized set cover problem in fixed parameter time.

On input a set $U = \{u_1, \ldots, u_m\}$, a family $S = \{S_1, \ldots, S_n\}$ of subsets $S_i \subseteq U$, and a positive integer k, $\mathcal{M}$ first computes the corresponding instance $f(U, S, k) = (P, N, k)$ of the parameterized consistency problem as described in the proof of Lemma 4. Then $\mathcal{M}$ runs the PAC-learning algorithm $\mathcal{A}$ with confidence parameter $\delta = 1/4$ and error parameter $\varepsilon = 1/(\|N\| + 2)$. For each request for a random classified example, $\mathcal{M}$ randomly chooses an example from $P \cup N$ and passes it to $\mathcal{A}$ along with its classification. After at most $g(k)poly(n, m)$ steps, $\mathcal{A}$ produces a boolean circuit computing some hypothesis h. Now $\mathcal{M}$ tries

to determine the relevant variables of h as follows. Observe that if h depends on at most k relevant variables, then for each relevant variable x_i and for a uniformly at random chosen assignment $x \in \{0,1\}^n$ we have $h(x) \neq h(x')$ with probability at least 2^{-k}, where x' is obtained from x by flipping the i-th bit. Thus, $\mathcal{M}$ can detect the index set I of all relevant variables of h with probability $\geq 3/4$ in time $poly(2^k, n)$, provided that h indeed depends on at most k variables (otherwise, I can be an arbitrary subset of $[n]$ and $\mathcal{M}$ might fail to find any relevant variables). Finally, $\mathcal{M}$ accepts if and only if $\|I\| \leq k$ and the monomial $\bigwedge_{i \in I} x_i$ is consistent with P and N.

Assume that (U, S, k) is a positive instance of the parameterized set cover problem. Then, by the choice of δ and ε, $\mathcal{A}$ produces with probability at least $3/4$ a k-junta h that is consistent with P and N. Now, using the properties of the instance (P, N, k) described in the proof of Lemma 4, it follows that if $\mathcal{A}$ is successful, then $\mathcal{M}$ finds with probability $\geq 3/4$ a monotone k-monomial consistent with P and N, implying that $\mathcal{M}$ accepts with probability $\geq 1/2$.

On the other hand, it is clear that $\mathcal{M}$ will never accept a negative instance (U, S, k). □

Thus it is rather unlikely that the class of k-monomials (or any other concept class considered in Theorem 3) is proper PAC-learnable in time $g(k)poly(n)$. In contrast, in the distribution-specific setting with respect to the uniform distribution, proper PAC-learning can be achieved in fixed parameter time. For the class of monotone k-juntas, this has already been shown by Mossel et al. [22].

Theorem 7. *Under the uniform distribution, k-monomials are PAC-learnable in deterministic fixed parameter time with k-monomials as hypotheses.*

Proof. Let f be some k-monomial and for any $i \in [n]$ consider the probability $p_i = \Pr[f(x) = x_i]$ for a uniformly chosen assignment $x \in \{0,1\}^n$. If x_i does not appear in f then $p_i = 1/2$. If x_i appears unnegated in f then $p_i = 1/2 + 2^{-k}$, and if x_i appears negated in f then $p_i = 1/2 - 2^{-k}$. The probability p_i can be estimated within additive error 2^{-k-1} with high probability by using $poly(2^k)$ random examples. Thus, we can successively determine all literals of f in time $poly(2^k, n)$. □

4 Learning k-Juntas Exactly

In this section we consider the parameterized learnability of concept classes that are evaluable in fixed parameter time. Our main result here is that any such class is randomized fpt EQ-learnable with access to an oracle in W[P], provided that the Hamming weight is used as parameter. Our learning algorithm uses a similar strategy as the exact learning algorithm of Bshouty et al. [7]. We first recall a version of the Valiant-Vazirani lemma [24] that lower bounds the probability that a randomly chosen linear function h isolates some $x \in D$ (we say that a function $h : \{0,1\}^s \to \{0,1\}^l$ *isolates* x in $D \subseteq \{0,1\}^s$, if x is the only string in D with $h(x) = 0^l$). Furthermore, it provides an upper bound on the probability that such an isolated x lies in a given small subset D' of D.

Lemma 8. *Let $D \subseteq \{0,1\}^s - \{0^s\}$ be a non-empty set of cardinality c, let $D' \subseteq D$ be of cardinality at most $c/12$, and let l be an integer such that $2^l < 3c \leq 2^{l+1}$. Then, for a uniformly chosen linear function $h : \{0,1\}^s \to \{0,1\}^l$,*

- *with probability at least $2/9$, there exists exactly one element $x \in D$ such that $h(x) = 0^l$, and*
- *with probability at most $1/18$, there exists some element $x \in D'$ such that $h(x) = 0^l$.*

Theorem 9. *Any* XP*-enumerable representation (R, κ) is randomized fpt EQ-learnable with access to a* uniform-XP *oracle and using boolean circuits as hypotheses. Moreover, if the Hamming weight is used as parameter, then a* W[P] *oracle suffices.*

Proof. We give an outline of the proof. Let $\hat{r}$ be the target. We describe a randomized learning algorithm $\mathcal{A}$ that on input n, $s = |\hat{r}|$ and $k = \kappa(\hat{r})$ collects a set S of counterexamples obtained from the teacher. To build a suitable hypothesis from the current set S, $\mathcal{A}$ randomly samples a polynomial number of concept names $r_1, \ldots, r_p$ from the set

$$Cons_{k,s}(S) = \{r \in R_{k,s} \mid R_n(r) \text{ is consistent with } S\}.$$

Then $\mathcal{A}$ makes an improper equivalence query using the hypothesis

$$\mathrm{maj}_{[r_1,\ldots,r_p]}(x) = \begin{cases} 1, & \|\{i \in \{1,\ldots,p\} \mid x \in R_n(r_i)\}\| \geq p/2, \\ 0, & \text{otherwise} \end{cases}$$

which is the majority vote on the concepts $R_n(r_1), \ldots, R_n(r_p)$. In order to do the sampling $\mathcal{A}$ will apply the hashing lemma stated above. More precisely, $\mathcal{A}$ cycles through all values $l = s, s-1, \ldots, 1$ and randomly chooses linear functions $h_i : \{0,1\}^s \to \{0,1\}^l$, $i = 1, \ldots, p$. Then $\mathcal{A}$ uses the oracle

$$B = \{(k, r, S, h, s, l) \mid \exists r' : rr' \in Cons_{k,s}(S) \text{ and } h(rr') = 0^l\},$$

where k is the parameter, to find for each function h_i a concept name r_i (if it exists) that is isolated by h_i in $Cons_{k,s}(S)$. Note that B belongs to uniform-XP as the representation (R, κ) is XP-enumerable. Now, for $i = 1, \ldots, p$ and each string $x \in \{0,1\}^n$ with the property that

$$\|\{r \in Cons_{k,s}(S) \mid x \in R_n(r) \Leftrightarrow x \in R_n(\hat{r})\}\| > (11/12)\|Cons_{k,s}(S)\|$$

(meaning that the inclusion of the counterexample x in S discards less than a $1/12$ fraction of all representations in $Cons_{k,s}(S)$) consider the random variable

$$Z_i(x) = \begin{cases} -1, & h_i \text{ isolates an } r_i \text{ in } Cons_{k,s}(S) \text{ with } x \in R_n(r_i) \Delta R_n(\hat{r}), \\ 0, & h_i \text{ does not isolate any string in } Cons_{k,s}(S), \\ 1, & h_i \text{ isolates an } r_i \text{ in } Cons_{k,s}(S) \text{ with } x \in R_n(r_i) \Leftrightarrow x \in R_n(\hat{r}). \end{cases}$$

Then, provided that l has the right value, it follows that

$$E(Z_i(x)) \geq (2/9 - 1/18) - 1/18 = 1/9$$

and by Hoefding's inequality we get

$$Prob\left[\sum_{i=1}^{p} Z_i(x) \leq 0\right] = 2^{-\Omega(p)}.$$

Since the equivalence query $h = \mathrm{maj}_{[r_1,\ldots,r_p]}$ only disagrees with the target on the classification of x if $\sum_{i=1}^{p} Z_i(x) \leq 0$, this means that with probability $1-2^{n-\Omega(p)}$, h allows only counterexamples x that discard at least a $1/12$ fraction of all representations in $Cons_{k,s}(S)$.

To complete the proof outline, note that it is easy to see that if we use the Hamming weight as parameterization, then the oracle B actually belongs to W[P]. □

As an immediate consequence we get the following corollary.

Corollary 1. *Any* XP*-enumerable representation* (R, κ) *is PAC-learnable in randomized fixed parameter time with access to a* uniform-XP *oracle. Moreover, if the Hamming weight is used as parameter, then a* W[P] *oracle suffices.*

By using the representation J of k-juntas described in Section 2.2, we immediately get the following positive learning result for k-juntas.

Corollary 2. *k-juntas are randomized fpt EQ-learnable with access to a* W[P] *oracle.*

Note that the hypotheses used by the query-learning algorithm can have up to n relevant variables. It is not hard to verify that this is essentially optimal for any algorithm with fixed parameter running time. To see this, suppose that $\mathcal{A}$ learns k-juntas with $g(k)n^c$ equivalence queries using circuits having at most l relevant variables as hypotheses. Consider the subclass D consisting of all monotone monomials with exactly k variables.

If $\mathcal{A}$ asks the constant $h \equiv 0$ function as an equivalence query, then no monotone monomial from D agrees with h on the counterexample $a = 1^n$. Otherwise let a be a counterexample such that $h(a) = 1$ where $a_i = 0$ on all positions i for which h does not depend on x_i. The number of hypotheses from D that agree with h on a is at most $\binom{l}{k}$. Hence, for every equivalence query h there is some counterexample a such that the algorithm $\mathcal{A}$ can discard at most $\binom{l}{k}$ hypotheses from D. By a simple counting argument it follows that

$$g(k)n^c\binom{l}{k} \geq \binom{n}{k} - 1,$$

implying that $l = \Omega(n^{1-c/k}/g(k)^{1/k})$. Thus it follows for all ε and for sufficiently large k and n that $l \geq n^{1-\varepsilon}/g(k)$.

We conclude this section with a remark on exactly learning a generalization of juntas with membership queries. Consider the natural generalization of k-juntas where the target f is a boolean function of k linear forms on the n variables over the field $\mathbb{F}_2$. More precisely, $f(x_1, \ldots, x_n) = g(a_1(x), \ldots, a_k(x))$, where each $a_i(x)$ is defined as a linear function $\sum_{j=1}^{n} a_{ij}x_j$ over $\mathbb{F}_2$, where $a_{ij} \in \{0, 1\}$. Using membership queries such "generalized" k-juntas are exactly learnable in time $2^{O(k)}n^{O(1)}$ by a direct application of the learning algorithm of Kushilevitz and Mansour [19, Theorem 5.3]. According to this result, a boolean decision tree of depth d and n variables with $\mathbb{F}_2$-linear functions at each node can be exactly learned with membership queries in deterministic time polynomial in n and 2^d. Now it suffices to observe that a generalized k-junta can be transformed into a decision tree of depth k with a linear function at each node.

5 Discussion and Open Problems

We have examined the parameterized complexity of learning k-juntas, with our notion of efficient learning as fixed parameter tractable learnability. Our main results are about the hardness of learning k-juntas and subclasses of k-juntas in the PAC model by reductions ¿from a W[2]-complete problem. On the other hand, as a consequence of a more general result we show that k-juntas are exactly learnable with improper equivalence queries and access to a W[P] oracle. Some interesting open questions remain.

The main open question is whether the learning result of [22] for k-juntas can be improved to show that k-juntas are fpt PAC-learnable with boolean circuits as hypotheses. A more modest question is whether (monotone) k-monomials are fpt PAC-learnable with boolean circuits as hypotheses having k' relevant variables, where $k' = g(k)$ only depends on k. From Theorem 6 we only know that if we choose for g the identity function, then this is not possible unless W[2] collapses. On the other hand, Warmuth's modification of Haussler's algorithm achieves PAC learning of k-monomials in polynomial time with $k \log(2/\varepsilon)$-monomials as hypotheses.

Acknowledgments

We thank the anonymous referees for their valuable comments.

References

1. Almuallim, H., Dietterich, T.G.: Learning boolean concepts in the presence of many irrelevant features. Artificial Intelligence 69(1-2), 279–305 (1994)
2. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Control 75, 87–106 (1987)
3. Blum, A.: My favorite open problems (and a few results). In: Talk given at 2001 NeuroCOLT Alpine Workshop on Computational Complexity Aspects of Learning, March 26-29,2001 Sestriere, Italy (2001)

4. Blum, A.: Learning a function of r relevant variables. In: COLT, pp. 731–733 (2003)
5. Blum, A., Rudich, S.: Fast learning of k-term DNF formulas with queries. Journal of Computer and System Sciences 51(3), 367–373 (1995)
6. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Occam's razor. Information Processing Letters 24(6), 377–380 (1987)
7. Bshouty, N., Cleve, R., Gavaldà, R., Kannan, S., Tamon, C.: Oracles and queries that are sufficient for exact learning. Journal of Computer and System Sciences 52, 421–433 (1996)
8. Bshouty, N., Tamon, C.: On the fourier spectrum of monotone functions. Journal of the ACM 43(4), 747–770 (1996)
9. Chvatal, V.: A greedy heuristic for the set covering problem. Mathematics of Operations Research 4(3), 233–235 (1979)
10. Downey, R.G., Evans, P.A., Fellows, M.R.: Parameterized learning complexity. In: Proc. 6th Annual ACM Conference on Computational Learning Theory, pp. 51–57. ACM Press, New York (1993)
11. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness I: Basic results. SIAM Journal on Computing 24(4), 873–921 (1995)
12. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999)
13. Fellows, M., Koblitz, N.: Fixed-parameter complexity and cryptography. In: Moreno, O., Cohen, G., Mora, T. (eds.) Proc. Tenth International Symposium on Applied Algebra, Algebraic Algorithms, and Error Correcting Codes. LNCS, vol. 673, pp. 121–131. Springer, Heidelberg (1993)
14. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006)
15. Haussler, D.: Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. Artificial Intelligence 36, 177–221 (1988)
16. Johnson, D.S.: Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences 9, 256–278 (1974)
17. Kearns, M.J., Vazirani, U.V.: An Introduction to Computational Learning Theory. MIT Press, Cambridge (1994)
18. Kolountzakis, M., Markakis, E., Mehta, A.: Learning symmetric juntas in time $n^{o(k)}$. Interface between Harmonic Analysis and Number Theory (2005)
19. Kushilevitz, E., Mansour, Y.: Learning decision trees using the fourier spectrum. SIAM Journal on Computing 22(6), 1331–1348 (1993)
20. Lipton, R., Markakis, E., Mehta, A., Vishnoi, N.: On the fourier spectrum of symmetric boolean functions with applications to learning symmetric juntas. In: Twentieth Annual IEEE Conference on Computational Complexity, pp. 112–119 (2005)
21. Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning 2, 285–318 (1988)
22. Mossel, E., O'Donnell, R., Servedio, R.P.: Learning juntas. In: Proc. 35th ACM Symposium on Theory of Computing, pp. 206–212. ACM Press, New York (2003)
23. Pitt, L., Valiant, L.G.: Computational limitations on learning from examples. Journal of the ACM 35(4), 965–984 (1988)
24. Valiant, L., Vazirani, V.: NP is as easy as detecting unique solutions. Theoretical Computer Science 47, 85–93 (1986)
25. Valiant, L.G.: A theory of the learnable. Communications of the ACM 27(11), 1134–1142 (1984)