Machine Learning, 41, 197–215, 2000

Adaptive Versus Nonadaptive Attribute-Efficient Learning*

PETER DAMASCHKE Peter.Damaschke@fernuni-hagen.de
Fern Universität, Theoretische Informatik II, 58084 Hagen, Germany

Editor: Lisa Hellerstein

Abstract. We study the complexity of learning arbitrary Boolean functions of n variables by membership queries, if at most r variables are relevant. Problems of this type have important applications in fault searching, e.g. logical circuit testing and generalized group testing. Previous literature concentrates on special classes of such Boolean functions and considers only adaptive strategies. First we give a straightforward adaptive algorithm using $O(r2^r \log n)$ queries, but actually, most queries are asked nonadaptively. This leads to the problem of purely nonadaptive learning. We give a graph-theoretic characterization of nonadaptive learning families, called r-wise bipartite connected families. By the probabilistic method we show the existence of such families of size $O(r2^r \log n + r^2 2^r)$. This implies that nonadaptive attribute-efficient learning is not essentially more expensive than adaptive learning. We also sketch an explicit pseudopolynomial construction, though with a slightly worse bound. It uses the common derandomization technique of small-biased k-independent sample spaces. For the special case $r = 2$, we get roughly $2.275 \log n$ adaptive queries, which is fairly close to the obvious lower bound of $2 \log n$. For the class of monotone functions, we prove that the optimal query number $O(2^r + r \log n)$ can be already achieved in $O(r)$ stages. On the other hand, $\Omega(2^r \log n)$ is a lower bound on nonadaptive queries.

Keywords: membership queries, relevant variables, nonadaptive learning, probabilistic method, group testing, monotone Boolean functions

1. Introduction

This paper addresses the problem of exact learning of Boolean functions by membership queries, provided that at most r of the n attributes (variables) are relevant. This is also known as attribute-efficient learning. We first introduce the learning model and then list our contributions. Formal definitions follow afterwards.

1.1. The learning model

Let f be a Boolean function, given as an oracle and initially unknown to the learner. The following procedure is called a membership query (cf. (Angluin, 1987)): The learner chooses an assignment x (where each of the variables gets one of the Boolean values 0,1) at his own discretion, and then the oracle delivers $f(x)$, i.e. the value of the function for this

*A preliminary version appeared in the *Proceedings of the 30th ACM Symposium on Theory of Computing (STOC'98)*, pp. 590–596.

assignment x. Exact learning means that the learner has to identify f uniquely. The goal is to learn f exactly, using a possibly small number of membership queries.

Trivially, 2^n queries are necessary to identify an arbitrary Boolean function with n variables. However, if the learner knows in advance that f belongs to some restricted class of Boolean functions then this knowledge may be used to devise a clever strategy that needs fewer queries.

We distinguish between adaptive and nonadaptive learning. In adaptive learning, the learner is allowed to choose his queries depending on the answers to all his previous queries. In nonadaptive learning, the learner must fix all queries he wishes to ask in advance, before obtaining any answer. (This means in particular that queries might be asked in any order, or simultaneously.) We also consider the intermediate setting of parallel learning. Here the learning process consists of a number of stages. In every stage, the learner chooses a set of queries which may depend on the answers obtained in all earlier stages, and then he asks them simultaneously. In particular, nonadaptive learning is parallel learning in one stage, whereas the number of stages is unbounded in the case of adaptive learning.

1.2. Our results

It is not surprising that our subject is connected to the well-known (n, r)-universal families. Informally, a family of assignments is called (n, r)-universal if it includes every assignment of every r-element subset of variables.

We start with a straightforward bound for adaptive learning of functions with r relevant variables, namely, $U(r, n) + r \log n$ queries are sufficient, where $U(r, n)$ denotes the minimum size of an (n, r)-universal family. For trivial reasons, $U(r, n)$ is a lower bound, and it is known from (Kleitman & Spencer, 1973) that $U(r, n) = \Omega(2^r \log n)$. Hence this upper bound is within a factor $1 + O(r/2^r)$ of optimal. (We mention that the case $r = 1$ is completely trivial: One can learn even nonadaptively by $\log n$ queries.) One can simply prove $U(r, n) = O(r2^r \log n)$[1] by standard application of the probabilistic method (see e.g. (Motwani & Raghavan, 1995)); deep results on $U(r, n)$ and further references can be found in Naor, Schulman, and Srinivasan (1995). Most importantly, there exist efficient deterministic constructions of (n, r)-universal families of only slightly larger size.

It should be noticed that the learner must know r (or a constant upper bound) before asking his membership queries: No efficient learning algorithm without previous knowledge of r can exist. Even the decision whether $r = 0$ or $r = n$ needs 2^n queries in the worst case (by a trivial adversary argument). Several restricted function classes can be efficiently learned by adaptive queries without prior knowledge of r, see e.g. (Blum, Hellerstein, & Littlestone, 1995; Bonis & Vaccaro, 1998; Triesch, 1996), some of them by a randomized strategy only (Damaschke, 1998c). In the case of nonadaptive learning it is anyhow impossible to get a good query bound in r and n, if r is not given to the learner.

We observe that the overwhelming majority of $U(r, n)$ queries can be asked nonadaptively. This introductory result provokes the question of nonadaptive learning and of query-stage tradeoffs, which leads us to the most interesting part.

Perhaps the main contribution is the combinatorial characterization of purely (one-stage) nonadaptive learning by r-wise bipartite connected families, and the subsequent complexity results. This adds a new meaningful type of combinatorial subset families (or assignment

families, codes, etc.) to the list of such structures. (For a general introduction to several combinatorial designs see e.g. (Colbourn & Dinitz, 1996).)

We prove the existence of r-wise bipartite connected families of size $O(r2^r \log n + r^2 2^r)$. This means that nonadaptive learning of functions with r relevant variables does not significantly exceed the complexity of adaptive learning. Using ideas from (Naor, Schulman, & Srinivasan, 1995), we also give a pseudopolynomial construction, though with an additional r factor in the size. A polynomial construction is missing.

For adaptive learning and particular small values of r it is interesting to improve the general upper bound $U(r, n) + r \log n$ and to aim at the best constant factors. We consider case $r = 2$ which is already nontrivial, unlike case $r = 1$. Exploiting a special $(n, 2)$-universal family and a previously known result on group testing with two defectives, we achieve $2.275 \log n + o(\log n)$ queries. (Note that $2 \log n$ is an obvious lower bound.) This also improves the results of (Uehara, Tsuchida, & Wegener, 1997) in case $r = 2$. The same question becomes less important for large r since the ratio of $U(r, n) + r \log n$ to $U(r, n)$ goes to 1, as mentioned earlier.

For the class of monotone functions with at most r relevant variables, the situation is quite different. It is a straightforward exercise to learn monotone functions by $O(2^r + r \log n)$ queries, and this is almost optimal. Here we prove that $O(r)$ stages are enough to keep this query number, even without prior knowledge of r. In contrast, learning monotone functions with r relevant variables nonadaptively needs $U(r, n)$ queries, and the learner must know r in advance. So there is an exponential gap in the factor before $\log n$, if we consider a range from $O(r)$ stages down to 1 stage. It remains open how this tradeoff looks in detail.

1.3. Related work

Beginning with Littlestone (1988), adaptive learning of Boolean functions with few relevant variables has been studied in Blum, Hellerstein, and Littlestone (1995), Bshouty and Hellerstein (1996), and Uehara, Tsuchida, and Wegener (1997) under several learning models. The focus was on special function classes that can be learned efficiently in the sense that the number of queries is polynomial in r, in $\log n$, and in the size of a representation. While the former papers develop a general theory, Uehara, Tsuchida, and Wegener (1997) provides tight results on the exact query number for specific classes. Approximate learning in the presence of many irrelevant attributes has been considered in Dhagat and Hellerstein (1994), and Kivinen, Mannila, and Ukkonen (1992); see also Almuallim and Dietterich (1994). Learning in an infinite attribute space (Blum, 1992) is a related topic.

Exact learning by membership queries can also be considered as interpolation of functions, based on function values at freely chosen points, which has been investigated for other restricted function classes, too, both in the adaptive and nonadaptive setting (Clausen et al., 1991; Roth & Benedek, 1991). Parallel learning has been studied in Bshouty and Cleve (1992), but not for the present problem. Fundamental issues of adaptive versus nonadaptive learning in the mistake-bound model are studied in Ben-David, Kushilevitz, and Mansour (1997), cf. also Goldman and Sloan (1994).

Structural characterizations and constructions of assignment families that learn special function classes with few relevant variables nonadaptively are given in Balding and Torney (1996) (disjunction) and Hofmeister (1999) (parity). In general, it is a fundamental topic

in concept learning to characterize assignment families which have certain learning power, cf. also Bshouty (1995) and Khardon and Roth (1996).

Another line of motivation is the well-known group testing problem, which is attribute-efficient learning of the disjunction of an unknown subset of variables. (Suppose that we want to find r out of n chemical samples which are contaminated with a certain substance, and we are able to test arbitrarily chosen subsets of samples for presence of the substance.) One main result about adaptive group testing is that $\log(\binom{n}{r}) + O(r)$ queries are sufficient, which misses the trivial lower bound only by the $O(r)$ term (Triesch, 1996). For some natural generalizations of group testing, e.g. to threshold testing, see Damaschke (1997, 1998c), Bonis and Vaccaro (1998), and Farach et al. (1997); the classes considered there lie between the disjunction and arbitrary functions with r relevant variables. (We mention that some results of Damaschke (1997) concerning threshold testing in a single sample have been subsequently improved in Damaschke (1998a) and Bonis, Gargano, and Vaccaro (1998).)

There are interesting applications of group testing in logic circuit checking (Seroussi & Bshouty, 1988), medical diagnosis (as suggested e.g. in Dhagat and Hellerstein (1994)), biological and chemical test series, and several other fields. Some of them are not only speculative, on the contrary, efficient learning strategies are really used in laboratories (Balding & Torney, 1995, 1996; Du & Hwang, 1993; Farach et al., 1997; Fischer, Klasner, & Wegener, 1999; Knill, 1995). For example, nonadaptive group testing is a highly interesting tool in DNA sequencing. Typical input sizes are $n > 1{,}000$ and $r < 10$. In this application nonadaptiveness is crucial, since the tests are time-consuming but can be performed in parallel. The restriction that r must be known before is not a serious obstacle; one often has a quite sure upper bound, due to experience or probability estimates.

The work of Beimel, Geller, and Kushilevitz (1998) addresses lower and upper bounds for the adaptive query complexity of learning several function clases, including t-term DNF.

1.4. Some terminology

A Boolean function of n variables (or attributes) is a mapping $f : \{0,1\}^n \to \{0,1\}$. For brevity we omit the adjective "Boolean". Let V be the set of variables. An assignment is a mapping $x : V \to \{0,1\}$, an assignment on a subset U of variables is defined similarly. An assignment y on U is called a restriction of another assignment x, or induced by x, if $y(u) = x(u)$ for each $u \in U$.

We will informally use denotations like $f(x, y, z, \ldots)$ where the "arguments" x, y, z are assignments of disjoint subsets of variables or of single variables. We also write $x \cup y$ to denote the composition of assignments x, y on disjoint subsets. The symbol 0 or 1 sometimes denotes the assignment on a subset where all variables have value 0 or 1, respectively. These laxities are convenient, the exact meaning will always be clear from the context.

If y is an assignment on U, the projection f_y is the function on $V \setminus U$ with $f_y(z) := f(y, z)$. If A is a family of assignments, A_y denotes the family of all restrictions of those assignments from A inducing y on U, to $V \setminus U$.

A Boolean function is monotone if $x \leq y$ (componentwise) implies $f(x) \leq f(y)$.

A variable v is called a relevant variable (RV for short) if it has a proper influence on the function value, i.e. there exists an assignment x on $V\backslash\{v\}$ such that $f(x,0) \neq f(x,1)$. Let $Rel(n,r)$ be the class of functions of n variables having at most r RVs.

Throughout the paper, log means $\log_2$. We always omit nonessential details such as ceiling brackets.

1.5. *Special assignment families*

It turns out that attribute-efficient learning is closely related to certain combinatorial families of assignments defined below.

Definition 1.1. A family A of assignments on V is called (n,r)-universal if every assignment on every subset of (at most) r variables is induced by some member of A. Let $U(r,n)$ denote the minimum size of an (n,r)-universal family of assignments on n variables.

These families are also called (n,r)-exhaustive in the literature. Next we introduce a new special type of (n,r)-universal assignment families.

Consider three mutually disjoint subsets $X, Y, Z \subset V$ with $X, Y \neq \emptyset$, and an assignment z on Z. (If $Z = \emptyset$ then z is the empty assignment.) For a fixed (n,r)-universal A, the bipartite graph $B(X,Y,z)$ is defined as follows: The vertices are all possible assignments x on X and y on Y, and the edge xy exists if and only if there is some assignment in A inducing $x \cup y \cup z$ on $X \cup Y \cup Z$.

The following definition is the central "invention" of this paper; its importance will become apparent in Section 3.

Definition 1.2. An (n,r)-universal family A is called r-wise bipartite connected if $B(X,Y,z)$ is connected for any three mutually disjoint subsets X, Y, Z with $X, Y \neq \emptyset$, $|X \cup Z| = |Y \cup Z| = r$, and for any assignment z on Z.

Note that none of the graphs $B(X,Y,z)$ has isolated vertices: Since A is (n,r)-universal, for any vertex x of $B(X,Y,z)$, there exists $a \in A$ inducing $x \cup z$ on $X \cup Z$. Furthermore, since a induces some y on Y, edge xy exists in $B(X,Y,z)$.

We will also need the observation that, in Definition 1.2, connectivity of our bipartite graphs is hereditary in the following sense:

Lemma 1.3. *Let A be r-wise bipartite connected, $X, Y, Z \subset V$ mutually disjoint subsets of variables with $X, Y \neq \emptyset$ and $|X \cup Z| = |Y \cup Z| \leq r$, and z an assignment on Z. Then $B(X,Y,z)$ is also connected.*

Proof: Consider any supersets $X^* \supset X$ and $Y^* \supset Y$ such that X^*, Y^*, Z^* are mutually disjoint, and $|X^* \cup Z| = |Y^* \cup Z| = r$. Then $B(X^*, Y^*, z)$ is connected. Moreover, $B(X,Y,z)$ is obtained from $B(X^*, Y^*, z)$ in the following way: For every assignment x on X, identify those vertices of X^* corresponding to assignments that induce x on X. Proceed similarly with Y^*. Two vertices x, y in $B(X,Y,z)$ are joined by an edge iff there is at least

one edge x^*y^* in $B(X^*, Y^*, z)$ where x^* induces x, and y^* induces y. From this it is obvious that connectivity will not be destroyed. □

The concept of an (A, f)-feasible set of variables will be useful in the proof of our characterization Theorem 3.1.

Definition 1.4. Consider a fixed function f and a fixed assignment family A. A subset $U \subseteq V$ is called (A, f)-feasible if all assignments $a \in A$ agreeing on U yield the same $f(a)$.

If A is (n, r)-universal, U is (A, f)-feasible, and $|U| \leq r$, then f induces a function $f_A[U]$ on U in a self-explanatory way.

2. Adaptive learning by universal assignment families

The central importance of universal families for our subject is based on the following facts. The obvious proofs are omitted.

Fact 2.1. *Let be $f \in Rel(n, r)$ and A some (n, r)-universal family. Then f is constant if and only if all $f(a)$, $a \in A$ are equal.*

Fact 2.2. *The worst-case complexity of testing whether a given $f \in Rel(n, r)$ is constant, is exactly $U(r, n)$. Consequently, any learning algorithm for this function class needs at least $U(r, n)$ queries in the worst case.*

On the other hand, the main work in learning such functions is to query some (n, r)-universal family, as we shall see next. The following is an immediate consequence of Definition 1.1.

Lemma 2.3. *Let A be (n, r)-universal and $S \subset V$ a set of $s < r$ variables. For any assignment x on S, the family A_x is $(r - s)$-universal on $V \setminus S$.*

Proof: Consider a set $T \subset V \setminus S$ of size $r - s$, and an arbitrary assignment y on T. Then the assignment $x \cup y$ on $S \cup T$ is induced by some member of A. Hence y itself is induced by some member of A_x. □

Further we need an obvious binary search routine for RVs.

Lemma 2.4. *Let f be an arbitrary function. Once we know assignments x and y with $f(x) \neq f(y)$, we can find some RV by $\log n$ further queries.*

Proof: Let Z be the set of variables where x and y disagree: $Z = \{v : x(v) \neq y(v)\}$. Clearly, there must exist an RV in Z. Split Z in two sets X and Y of approximately the half size. We define an assignment z as follows: $z(v) := x(v)$ if $v \in X$, $z(v) := y(v)$ if $v \in Y$,

and $z(v) := x(v) = y(v)$ if $v \in V \setminus Z$. Note that z and x disagree on Y, and similarly, z and y disagree on X. Query $f(z)$. In either case ($f(z) \neq f(x)$ or $f(z) \neq f(y)$) we have found a pair of assignments disagreeing on $\lceil |Z|/2 \rceil$ variables only. After $\log n$ repetitions of this construction, one RV is identified. □

Now we can prove an upper bound:

Theorem 2.5. *Functions from $Rel(n, r)$ can be learned by $U(r, n) + r \log n$ queries.*

Proof: Fix some (n, r)-universal family A, and query $f(a)$ for all $a \in A$. If we take a minimum A, these are $U(r, n)$ queries. If all responses are equal then f is constant by Fact 2.1, and we are done. Otherwise we have found two assignments with different function values, and we identify some RV by $\log n$ further queries, due to Lemma 2.4.

Now assume that a set S of s RVs is already identified. We will show that, due to the informations gained from A, we either find some further RV by $\log n$ queries, or guarantee the nonexistence of more RVs without further queries. Together this proves our upper bound: Note that the 2^r assignments on the RVs are already contained in A, hence if we know the RVs then we have learned f, too.

Split A into the subfamilies A_x, according to the assignments x on S. By Lemma 2.3, every A_x is $(r - s)$-universal on $V \setminus S$. Note that $f_x \in Rel(n - s, r - s)$, so Fact 2.1 yields that the projection f_x is constant if and only if all values on A_x are equal. If all f_x are constant then no RV can exist outside S. In the other case we have assignments x on S and y, z on $V \setminus S$ such that $f(x, y) \neq f(x, z)$. Applying Lemma 2.4 to f_x we find an RV which is not in S. This completes the proof. □

Since $U(r, n) = \Omega(2^r \log n)$ ((Kleitman & Spencer, 1973), see also Naor, Schulman, and Srinivasan (1995)), Fact 2.2 and Theorem 2.5 together imply:

Corollary 2.6. *For any fixed $r \geq 2$, the complexity of learning functions from $Rel(n, r)$ is $c(r)U(r, n)$, where $\lim_{r\to\infty} c(r) = 1$.*

Note that our learning algorithm mainly consists of a nonadaptive phase (zero-testing) where A is scanned, followed by r adaptive search phases where the queries depend on previous answers. Later we will show for $r = 2$ that the adaptive part can be reduced considerably, just by better exploitation of the information gained from A. (For large r, this is less interesting; see the discussion in Section 1.2.)

An open problem is whether the r search threads in Theorem 2.5 can be parallelized. This would imply that roughly $\log n$ (rather than $r \log n$) stages are enough, without deteriorating the total number of queries, and would therefore be interesting for time-consuming applications. The difficulty is that several threads may find the same RV.

3. Nonadaptive learning

In this section we give an upper bound for the complexity of purely nonadaptive learning of functions $f \in Rel(n, r)$. That is, we construct an a-priori family A of assignments such

that the values of f on A enable us to infer the RVs and f. Although logarithmically, our result is somewhat worse than in the adaptive case.

First of all, A must be (n, r)-universal, otherwise we could not even decide whether f is constant, see Fact 2.2.

3.1. The characterization theorem

Theorem 3.1. *Let A be some fixed (n, r)-universal family on V (with $n = |V| \geq 2r$). Then A learns functions from $Rel(n, r)$ nonadaptively (i.e. the values $f(a)$, $a \in A$ are sufficient for identifying f, provided that $f \in Rel(n, r)$) if and only if A is r-wise bipartite connected.*

Proof: "*only if*". Assume that r is not r-wise bipartite connected. We have to construct functions $f, f' \in Rel(n, r)$ such that $f \neq f'$ but $f(a) = f'(a)$ for all $a \in A$. That means, A cannot distinguish between these functions and is not suitable for nonadaptive learning.

There exist mutually disjoint sets $X, X', Z \subset V$, such that $|X \cup Z| = |X' \cup Z| = r$, and $B(X, X', d)$ is disconnected for some assignment d on Z. We fix a function g on $X \cup Z$ having the following properties:

- For all assignments x on X belonging to the same connected component of $B(X, X', z)$, all $g(x, z)$ are equal.
- Both function values 0 and 1 appear among the $g(x, d)$.

Let f be the extension of g, i.e. that function on V without RVs outside $X \cup Z$, such that $f_A[X \cup Z] = g$. Trivially, $X \cup Z$ is (A, f)-feasible.

Let g' be the unique function on $X' \cup Z$ such that, in every connected component of every $B(X, X', z)$, all function values of g *and* g' are equal. By the remark after Definition 1.2, none of the graphs $B(X, X', z)$ contains isolated vertices, hence g' is well-defined. Finally let f' be the extension of g'. i.e. that function on V without RVs outside $X' \cup Z$, such that $f'_A[X' \cup Z] = g'$.

Consider any $a \in A$. It induces assignments on X, X', Z, say x, x', z, respectively. By definition, there is an edge xx' in $B(X, X', z)$. Since x and x' lie in the same connected component, our construction of g' yields $g(x, z) = g'(x', z)$. Since f and f' are just extensions of g and g', respectively, we have $f(a) = f'(a)$.

On the other hand, by our choice of g there exist assignments x_1, x_2 on X with $g(x_1, d) \neq g(x_2, d)$. Moreover, since A is (n, r)-universal, we have assignments $a_1, a_2 \in A$ inducing x_1, x_2 on X and d on Z, respectively. Together this means $f(a_1) \neq f(a_2)$. Since a_1, a_2 agree on Z, function f has some RV outside Z. Clearly, this RV must be in X. But f' has no RV in X, so we conclude $f \neq f'$.

"*if*". $B(X, Y, z)$ is connected for any three disjoint sets X, Y, Z with $|X \cup Z| = |Y \cup Z| = r$ and any assignment z on Z. By Lemma 1.3 this remains true if $|X \cup Z| \leq r$ and $|Y \cup Z| \leq r$.

Consider any function $f \in Rel(n, r)$. We have to show that f can be uniquely identified from the responses $f(a)$, $a \in A$. It suffices to determine the set R of RVs of f. Since A is (n, r)-universal, it induces in particular all possible assignments on R, thus we can learn f.

Let X, Y, Z be variable sets as above, having the additional property that both $X \cup Z$ and $Y \cup Z$ are (A, f)-feasible. We claim that $f_A[X \cup Z](x, z)$ is independent of x, in other words, Z is (A, f)-feasible, too.

To see this, consider two assignments x_1, x_2 on X. There exist $a_1, a_2 \in A$ inducing x_1, x_2 on X and z on Z, respectively. Let y_1, y_2 be the assignments on Y induced by a_1, a_2, respectively. Then $B(X, Y, z)$ contains the edges x_1y_1 and x_2y_2. Since the graph is connected, there exists a path starting and ending in these two edges. By the definition of $B(X, Y, z)$, every edge is created by some assignment from A. If we switch from an edge xy of our path to an incident edge, say $x'y$, we also switch from some $a \in A$ inducing $x \cup y \cup z$ on $X \cup Y \cup Z$ to some $a' \in A$ inducing $x' \cup y \cup z$. Since $Y \cup Z$ is (A, f)-feasible, we have $f(a) = f(a')$. The other case (both edges have a common vertex in Y) is symmetric; remember that $X \cup Z$ is also (A, f)-feasible. So we get $f(a_1) = f(a_2)$ by straightforward induction on the path. But this means $f_A[X \cup Z](x_1, z) = f_A[X \cup Z](x_2, z)$. So (A, f)-feasibility of sets of at most r variables is closed under intersection. Consequently, the intersection F of all (A, f)-feasible sets of size at most r is the unique smallest (A, f)-feasible set.

Finally we claim that $F = R$. Since we can recognize the (A, f)-feasible sets from the $f(a)$, $a \in A$ alone, this completes the proof.

Trivially, R is (A, f)-feasible, hence $F \subseteq R$. Assume that $F \subset R$ is a proper inclusion. Choose some $v \in R \backslash F$. Since v is an RV, there exists an assignment z on $R \backslash \{v\}$ with $f_A[R](z, 0) \neq f_A[R](z, 1)$. Since A is (n, r)-universal, there exist assignments in A inducing $(z, 0)$ and $(z, 1)$ on R. In particular, they agree on $F \subseteq R \backslash \{v\}$. This contradicts the (A, f)-feasibility of F. □

Corollary 3.2. *We can learn functions from $Rel(n, r)$ by $U(n, 2r) = O(r2^{2r} \log n)$ nonadaptive queries.*

Proof: Every $2r$-universal family is r-wise bipartite connected, namely, they yield complete bipartite graphs $B(X, Y, z)$. □

However, completeness is a much stronger property than connectivity, and indeed, the factor $r2^{2r}$ is poor. Next we aim at better estimates.

3.2. Constructions of r-wise bipartite connected families

We invoke the probabilistic method to prove the existence of r-wise bipartite families being not much larger than (n, r)-universal families. We start with some technical lemmas.

Lemma 3.3. *Consider a bipartite graph with m vertices in each part, and with $k > 1$ connected components. If we add a random edge then the number of connected components decreases with probability at least $k/m - k^2/4m^2$.*

Proof: We claim that the worst case appears if all components but one are isolated vertices and both parts of the bipartite graph contain about $k/2$ of them. Then the good event is that our random edge meets at least one isolated vertex. This gives the asserted probability.

To prove the claim, we use the following notion. A connected component is said to be of size (x, y) if it has x vertices in one part and y vertices in the other part. Consider any two connected components of size (a, b) and (c, d). There are $ad + bc$ possibilities to join these components by a further edge. If the sizes are $(a - 1, b)$ and $(c + 1, d)$ instead, we have $ad + bc + b - d$ such possibilities. If $b < d$ then the probability that a further random edge joins the components is smaller in the latter case. By an obvious inductive argument we conclude that the worst case consists of one big component along with $k - 1$ isolated vertices.

If there are j and $k - j$ isolated vertices, respectively, in both parts then we have $jm + (k - j)m - j(k - j) = km - j(k - j)$ possibilities to join two components. This is minimized for $j = \lfloor k/2 \rfloor$. □

Lemma 3.4. *Consider the following random experiment*: *Start with a bipartite graph with m vertices in each part, and empty edge set. Add t random edges independently with repetition, i.e. in each step, take one of the* m^2 *possible edges equiprobably. Then after* $\Theta(m \log^2 m)$ *initial steps, the probability that the bipartite graph is still disconnected after t further steps behaves as* $O((1 - 3/4m)^t)$.

Proof: We observe the number k of connected components, descending from $2m$ to 1 during the process. Instead of the given experiment we consider a slower process, namely the Markov chain M with states $k = 2m, \ldots, 1$, such that $k \geq 2$ becomes $k - 1$ with probability $k/2m$ and remains k otherwise. (In particular, all states $k \geq 2$ are transient.) Note that $k/m - k^2/4m^2 \geq k/2m$. We shall prove that, in this Markov chain, the probability of $k > 1$ behaves as asserted. Clearly, this result and Lemma 3.3 together imply the assertion for the original experiment, too.

For the sake of simplicity we will further slow down our process: Initially we make all states c except $c = 2m$ "inactive" Later we activate $c = 2m - 1, 2m - 2, \ldots, 3, 2$ as described below. The meaning is the following: An inactive state c remains c with probability 1. Only after c has been activated, the probability of transition from c to $c - 1$ becomes $c/2m$. Hence, as soon as $c = 2$ has become active, the transition probabilities are the same as in M. We shall prove that even in this slower process, the probability of $k > 1$ behaves as asserted, if we choose the activation times appropriately.

In the first step we get $k = 2m - 1$ with probability 1. For any state c and any moment, we define the rate of loss at c to be the conditional probability that $k < c$ after the next step, under the condition that $k \geq c$ before the step. By downwards induction on $c = 2m - 1, \ldots, 2$ we will show:

Claim. *If state c is activated* $\Theta(m(\ln c + \ln 2)/c)$ *steps later than* $c + 1$ *then, after activation of c, the rate of loss at c remains at least* $(c - 1/2)/2m$ *forever*.

This is trivial for $c = 2m - 1$: After 1 step, the rate of loss is constantly $(2m - 1)/2m > (c - 1/2)/2m$.

For the inductive step from $c + 1$ to c consider the time when $c + 1$ is activated. By the inductive hypothesis, the rate of loss at $c + 1$ is at least $(c + 1/2)/2m$ further on. Thus we may consider the states above c as one superstate with transition probability $(c + 1/2)/2m$.

Hence, after $\Theta(m(\ln c + \ln 2)/c)$ steps, we have $k > c$ only with probability at most $1/2c$: Note that

$$\left(1 - \frac{c + 1/2}{2m}\right)^{2m/(c+1/2)} < 1/e,$$

hence $\ln(2c)$ repetitions reduce the probability to $1/2c$, and $\frac{2m}{c+1/2} \ln(2c)$ may also be expressed as $\Theta(m(\ln c + \ln 2)/c)$. If we activate state c now, the rate of loss at c is at least $(1 - 1/2c)c/2m = (c - 1/2)/2m$ at this moment. Namely, we have $k = c$ with probability at least $(1 - 1/2c)$, and $c/2m$ was the transition probability to the next state $c - 1$.

Since the rate of loss at $c+1$ is larger, the ratio of probabilities of $k > c$ and $k = c$ further decreases in time. (This can be verified by a short calculation.) It follows that the rate of loss at c will not fall below $(c - 1/2)/2m$ in the future. This completes the induction step.

The lemma follows if we apply the claim to $c = 2$ and sum up the activation times over all c. □

Theorem 3.5. *There exists an r-wise bipartite connected family on n variables, consisting of $O(r2^r \log n + r^2 2^r)$ assignments.*

Proof: Consider a family A of size t, where we simply assign 0 or 1 with probability $1/2$ to each of the nt bits of A, independent of each other. We prove that all bipartite graphs addressed by Theorem 3.1 are connected with some positive probability, for $t = \Theta(r^2 2^r + r2^r \log n)$.

Consider mutually disjoint subsets $X, Y, Z \subset V$ with $|Z| = s$ and $|X| = |Y| = r - s$ $(0 \le s \le r)$, and an assignment z on Z. Due to the construction of A, $B(X, Y, z)$ is a random bipartite graph in the sense of Lemma 3.4, with some modification: In case $s > 0$, a new edge is added only if the assignment induced on Z is z, that is, we have a "time dilatation" by a factor 2^s. Note that A becomes r-wise bipartite connected (this includes (n, r)-universality) as soon as

- all assignments on sets of size r have appeared and thus all vertices of all $B(X, Y, z)$ are created, and
- all $B(X, Y, z)$ for $s < r$ are connected.

For any $s < r$, the number of bipartite graphs to be considered is generously bounded by $n^s n^{r-s} n^{r-s} = n^{2r-s}$. We apply Lemma 3.4 with $m = 2^{r-s}$. So the probability of disconnectivity behaves as $O((1 - (3/4)2^{s-r})^{t/2^s})$, after $\Theta((r - s)^2 2^{r-s})$ initial steps and t further steps. (Note that the last term depends on r but not on n.) If we choose $t = \Omega(r^2 2^r)$ such that, for all $s < r$, the product $n^{2r-s}(1 - (3/4)2^{s-r})^{t/2^s}$ is below some appropriate small constant then A fulfills the criteria with positive probability.

Routine calculation shows that $t = ar2^r \log n$ is sufficient, where a is independent of r and n. We get

$$n^{2r-s}(1 - (3/4)2^{s-r})^{ar2^{r-s} \log n} = n^{2r-s+ar2^{r-s} \log(1-(3/4)2^{s-r})}.$$

It suffices to keep the exponent negative. Since $\ln(1-x) < -x$, we can upperbound the exponent by $(2 - 3a/4\ln 2)r - s$, and we are done. □

So we obtain an r-wise bipartite connected family simply by adding random assignments until all conditions are satisfied. However, for problems of this type one is always interested in explicit deterministic constructions, too.

As already mentioned, any $(n, 2r)$-universal family is r-wise bipartite connected by itself. However, any explicit construction of r-wise bipartite connected families via $(n, 2r)$-universal families would yield a factor at least 2^{2r}. Considerable progress is made by the next theorem.

Theorem 3.6. *For any r, an r-wise bipartite connected family of size $O(r^2 2^r \log n)$ can be constructed in $O(g(r)n^{2r+1})$ time (with g being some exponential function of r).*

Proof: We follow the lines of Theorem 1 in Naor, Schulman, and Srinivasan (1995). This approach works although our problem does not fit into the framework of so-called r-restriction problems in the sense of Naor, Schulman, and Srinivasan (1995). (We omit the definition of this concept here; it is not necessary for understanding the proof.) We can apply the method after decomposing our problem strictly into a sequence of r-restriction problems. This straightens the situation, but incurs an additional r-factor.

The main tool is an ϵ-biased $2r$-wise independent sample space S of assignments on n variables; such structures of size $poly(r \log n/\epsilon)$ can be constructed efficiently and deterministically (Alon et al., 1992a; 1992b; Naor & Naor, 1993) . For any $s \leq 2r$, the probability that a random assignment induces a prescribed assignment on a fixed subset of size s is $2^{-s} \pm \epsilon$.

We want to choose a subfamily $A \subseteq S$ that

(1) makes all bipartite graphs $B(X, Y, z)$ (as in Theorem 3.1) with $|Z| < r$ connected, and
(2) induces all possible z on Z for $|Z| = r$.

Clearly, there are less than n^{2r} such bipartite graphs with initially at most 2^{r+1} connected components each. Since we aim at an upper bound anyhow, we pretend for simplicity that our n^{2r} bipartite graphs have exactly 2^r vertices in both parts, which is the worst case. By this, we also need not verify condition (2) separately.

Suppose that we have already chosen a subfamily of assignments and take a new random assignment from a $2r$-wise independent sample space. This means the addition of a random edge in each $B(X, Y, z)$. Applying Lemma 3.3 with $m = 2^r$ we would reduce the number of components from k to $k-1$ with probability at least $k/2^r - k^2/2^{2r+2}$. In case of an ϵ-biased sample space however, the guaranteed probability of decreasing the number of connected components is somewhat smaller, but at least $k/2^r - k^2/2^{2r+2} - 2^{2r}\epsilon$. (In the worst case, each of the edges decreasing the number of connected components might be chosen with probability decreased by ϵ.) Let us fix $\epsilon = 2^{-4r}/12$. Then the probability of success remains larger than $k/2^r - (k/2^r)^2/3$, and S has size $poly(2^{2r} \log n)$. (We remark that this is larger than our asserted upper bound for $|A|$, therefore we must not simply take $A = S$ now.)

We construct A by successively taking assignments from S, due to some rule specified below, until all graphs $B(X, Y, z)$ become connected. For simplifying the analysis we make a further (and more drastic) aggravating assumption: As long as there remain graphs with k connected components, all graphs having already $k-1$ connected components "wait" for them, i.e. edges added in these graphs which further reduce the number of components are ignored. This separates the process into phases $k = 2^{r+1}, \ldots, 4, 3, 2$ where all graphs reduce their component number from k to $k-1$. (Note the similarity to the "slow-down" in the proof of Lemma 3.4.)

Consider any fixed k. By the pigeonhole principle, there exists an assignment in S which reduces the number of connected components in a $k/2^r - (k/2^r)^2/3$ fraction of all bipartite graphs having still k components. This assignment can be naively found and checked by exhaustive search in S; note that the total time is dominated by $O(n^{2r+1})$ (subject to some factor independent of n) if we estimate polylog(n) by n.

Now we estimate the number of steps after which all graphs are ready for the next phase. For $k \geq 2^{r-1}$, we always meet at least a constant fraction of our set of bipartite graphs, thus we need $O(r \log n)$ assignments in phase k. This is a total of $O(r2^r \log n)$ assignments in $O(2^r)$ phases. For $k < 2^{r-1}$, the success probability is larger than $k/2^{r+1}$. Thus the duration of phase k is bounded by

$$O\left(\frac{\log n^{2r}}{\log(1 + k/2^{r+1})}\right) = O(k^{-1} r 2^r \log n).$$

Since the sum of the harmonic series up to $k = 2^r$ is $O(\log k) = O(r)$, this finally yields $|A| = O(r^2 2^r \log n)$. □

We conjecture that we can get rid of the second r factor by relaxing our strict decomposition into phases and monitoring the component numbers of the several bipartite graphs in a pipelined fashion. The difficulty is to formulate a suitable rule of choice of the next assignment from S. Possibly, an approach similar to those used in searching with lies (pebbles, weight function etc.) might be successful.

Note that the above construction is only pseudopolynomial. An explicit construction with a polynomial time bound in both n and r, as it is known for (n, r)-universal families and other r-restriction problems, is missing. We conjecture that the splitter method developed in Naor, Schulman, and Srinivasan (1995) can attack this problem.

Moreover, concrete factors for the first r would be interesting. Case $r = 1$ is completely trivial, but already case $r = 2$ is a challenge.

3.3. Remark: Auxiliary computations

In this paper we considered the pure query complexity only. However, for applying a nonadaptive learning strategy we must explicitly compute the set R of RVs from the values $f(a)$ ($a \in A$), that is, the smallest (A, f)-feasible set. Naive exhaustive search would require more than n^r computations. Nevertheless there exists a practicable solution to this problem which drastically reduces the amount of auxiliary computation. We refer to Damaschke (1998b).

4. Monotone functions

It is straightforward to learn monotone functions by $O(2^r) + r \log n$ adaptive queries without any previous knowledge about r, using r binary search procedures, and for simple information-theoretic reasons this cannot be improved. Interestingly, $O(2^r + r \log n)$ queries are sufficient even if we learn "almost" nonadaptively. More precisely, the stage number is independent of n.

Before proving this, let us introduce some further notion. For convenience we will loosely identify an assignment with its 1-set, i.e. the set of variables having value 1 in this assignment.

In the course of a learning process, a subset $A \subseteq V$ of the variables is called "interesting" if we know an assignment z on $\bar{A} = V \backslash A$ such that $f(z, 0) = 0$ and $f(z, 1) = 1$ (where arguments 0 and 1 denote the all-0 and all-1 assignment on A). Such z will be called a residual assignment. Clearly, an interesting subset contains an RV.

Theorem 4.1. *Monotone functions from $Rel(n, r)$ can be learned in $O(r)$ stages, using a total of $O(2^r + r \log n)$ queries. It is not necessary to know r in advance.*

Proof: Ask $f(\emptyset)$ and $f(V)$. Since f is monotone, we know that f is constant if these values are equal. So assume that $f(\emptyset) = 0$ and $f(V) = 1$. That means, set V is interesting.

Let us represent the variables by binary words $b_1 \cdots b_k$ of length $k = \log n$. If n is not a power of 2, we may add dummy (non-RV) variables. In the following, B_i^c ($c \in \{0, 1\}$) denotes the set of variables with $b_i = c$, or the assignment with 1-set B_i^c. Ask simultaneously the $2 \log n$ assignments B_i^c.

Assume that we find some i with $f(B_i^0) = f(B_i^1)$. Since $f(\emptyset) = 0$ and $f(V) = 1$, we see immediately that both B_i^0 and B_i^1 contain an RV. Consequently, both B_i^0 and B_i^1 are interesting.

It remains the case that $f(B_i^0) \neq f(B_i^1)$ for all i. By renaming, we may w.l.o.g. assume that $f(B_i^c) = c$ for all i, c. (Whenever $f(B_i^c) \neq c$, we switch the i-th bit in every word.)

Now define $y_i = \bigcap_{j=1}^{i} B_j^1$ for $i > 0$, and $y_0 = V$. Note that y_k contains a single variable. Query simultaneously all y_i. Since f is monotone, we have $f(y_i) \geq f(y_{i+1})$. Remember that $f(y_0) = 1$. If still $f(y_k) = 1$ then the only variable in y_k is an RV, because of $f(\emptyset) = 0$.

Otherwise there exists i with $f(y_i) = 1$ and $f(y_{i+1}) = 0$. We see immediately that $y_i \backslash y_{i+1}$ contains an RV. Moreover we know a residual assignment, hence this subset is interesting. Since $f(B_{i+1}^0) = 0$, $y_i \backslash y_{i+1} \subseteq B_{i+1}^0$, and f is monotone, we have $f(y_i \backslash y_{i+1}) = 0$. From $f(y_i) = 1$ we see now that y_{i+1} also contains an RV. Moreover we know a residual assignment, so y_{i+1} is interesting, too.

Let us resume: Either we have isolated some RV in one stage, or we have found two disjoint interesting subsets of a formerly known interesting set in two stages. Every stage performs at most $2 \log n$ queries.

Using the residual assignments we can continue the above process recursively on disjoint interesting sets, in an obvious way. Thus we produce a binary tree of interesting sets, where each inner node has two sons and each leaf contains a detected RV.

After this we check whether $f(z, 0) < f(z, 1)$ holds for some assignment z on the set S of RVs detected so far. If there is no such z then f has no further RVs; this is clear from

monotonicity of f. Otherwise we have an interesting subset $V \setminus S$, along with a residual assignment z. So we can continue the above process. Note that we must check only such assignments z not queried before, hence the total number of such initialization queries in the whole algorithm is bounded by 2^{r+1}.

Altogether we produce a sequence of binary trees as explained above. Trivially, the total number of nodes (stages) is $O(r)$, hence we used $O(2^r + r \log n)$ queries. □

We conjecture that the stage number can be further reduced without aggravating the query number, since the worst case (every tree has only one node) seems very sporadic. However 1 stage is not enough, due to

Fact 4.2 *Let r be arbitrary but fixed. Any assignment family that learns monotone functions from $Rel(n, r)$ nonadaptively must be (n, r)-universal and so it has size $\Omega(2^r \log n)$.*

Proof: Let A be a family that lacks of (n, r)-universality. Then some assignment z on a set R of size r is not induced by any member of A. We construct distinct monotone functions f, f' with all RVs in R, that cannot be distinguished by A. We may consider monotone functions g, g' on R instead. Let $g(z) = 0$ and $g(y) = 1$ for all $y > z$. We define g' by $g'(z) = 1$ and $g' = g$ else. □

So there is a considerable difference between the adaptive and nonadaptive query number, in contrast to the case of arbitrary functions in $Rel(n, r)$.

5. Adaptive learning with two relevant variables

In group testing, the case $r = 2$ has received special interest, cf. Althöfer and Triesch (1993), Damaschke (1994), Triesch (1996), and Macula and Reuter (to appear). The present section is devoted to the more general problem of learning Boolean functions in $Rel(n, 2)$. In the introduction we argued that this is a particularly interesting case.

Lemma 5.1. *We have* $\log n \leq U(2, n) \leq \log n + o(\log n)$.

Proof: The lower bound is obvious. The problem of constructing a good $(n, 2)$-universal family is essentially equivalent to finding a good Sperner family; for completeness we sketch a proof of the upper bound. Choose an even positive integer k such that $\binom{k}{k/2} \geq n$ and represent the n elements $v \in V$ by mutually distinct binary code words $v_1, \ldots, v_k$ where $v_i = 1$ for exactly $k/2$ places i. Let A be the family of assignments containing the constant assignments 0, 1 and, for each i $(1 \leq i \leq k)$, the assignment x_i with $x_i(v) = v_i$. Clearly, A is $(n, 2)$-universal. Routine calculations with help of Stirling's formula show that k as asserted above is sufficient. □

Theorem 2.5 and Lemma 5.1 imply that functions from $Rel(n, 2)$ can be learned by $3 \log n + o(\log n)$ queries, but we can do considerably better. Note that there exist (up to

symmetries, negations, and duality) only a few types of Boolean functions f with at most two RVs u, v, namely: 0, u, $u \wedge v$, $u \oplus v$, $u \wedge \neg v$.

Theorem 5.2. *Functions $f \in Rel(n, 2)$ can be learned by $2 \log n + o(\log n)$ queries, if f is not of type $u \wedge \neg v$. Otherwise, $2.275 \log n + o(\log n)$ queries are sufficient.*

Proof: First query $f(0)$ and $f(1)$. If $f(0) = 1$ then we may consider $\neg f$ instead of f. Thus let us assume $f(0) = 0$, without loss of generality. Since we have $r = 2$, if $f(1) = 1$ then f is monotone. Hence we can learn f by $2 \log n + O(1)$ queries in this case (see Section 4). So assume $f(1) = 0$ in the following. Then we know that f is either $u \oplus v$ or $u \wedge \neg v$. (Since we may rename u and v, the latter case includes $v \wedge \neg u$.)

Query the assignments of the special family A of size k, as defined in Lemma 5.1. From Fact 2.1 we know that f is constantly 0 if and only if all responses are 0. Otherwise there exists some j where $f(x_j) = 1$. Query the complement of x_j. If the response is 1 then f is of type $u \oplus v$, otherwise f is of type $u \wedge \neg v$.

Let f be of type $u \oplus v$. Then a pair (u, v) is a candidate for the pair of RVs if $u_i = v_i$ whenever $f(x_i) = 0$, and $u_i \neq v_i$ whenever $f(x_i) = 1$. Hence each variable u occuring in the candidate pairs has a unique partner v. In other words, the candidate pairs are disjoint. Now we find the proper pair in a straightforward way by $\log n$ further adaptive queries.

The $u \wedge \neg v$ case is more cumbersome. Let m denote the cardinality of set $\{i : f(x_i) = 1\}$. The ordered candidate pairs (u, v) are now characterized by the following condition: $u_i > v_i$ if and only if $f(x_i) = 1$. This also implies $m \leq k/2$. We see that no variable can be a candidate for both u and v, hence the candidate pairs form the edge set of some bipartite digraph whose vertices are the variables. Let us upperbound the number E of such pairs.

Consider a $k \times n$ matrix whose columns are the binary code words from Lemma 5.1; note that every column contains $k/2$ symbols 0 and 1, respectively. The rows (which correspond to the assignments in A) may be permuted such that the first m rows yield $f = 1$. A candidate pair (u, v) corresponds to a pair of columns. The u column must contain exactly $k/2 - m$ symbols 1 in the last $k-m$ positions. There are $\binom{k-m}{k/2-m}$ possibilities. Similarly, the v column contains exactly $k/2-m$ symbols 0 in the last $k-m$ positions, but combination with symbol 1 from u is forbidden in these rows. Thus each u has at most $\binom{(k-m)-(k/2-m)}{k/2-m}$ partners v. Therefore we have:

$$E \leq \frac{(k-m)!}{m!(k/2-m)!^2}.$$

Consider an arbitrary assignment y. A response $f(y) = 1$ keeps exactly those candidate pairs (u, v) where $y(u) = 1$ and $y(v) = 0$. It is important to observe that this is the edge set of some *induced* bipartite subgraph of the previous candidate graph; the other edges are discarded. A response $f(y) = 0$ keeps the other candidate pairs. Hence we have nothing else than the group testing problem on bipartite graphs which can be solved by $\log E + 1$ queries. (Here we do not need the result of Triesch (1996) in full generality; for bipartite graphs there exists a quite simple and elegant group testing strategy (Althöfer & Triesch, 1993).) It remains to upperbound $k + \max_{m \leq k/2} \log E$.

It may be instructive to see first a simple argument that E is subquadratic: As already mentioned, (u, v) is a candidate iff the induced assignment is $(1, 0)$ whenever the function value is 1, and one of the three other pairs else. Hence E is trivially bounded by $3^{\log n + o(\log n)} = n^{\log 3 + o(1)}$.

For a closer estimation we make use of Stirling's formula again. By the subadditivity of log we can drop the minor terms and get:

$$\log E < (k-m)\log(k-m) - m\log m - 2(k/2-m)\log(k/2-m).$$

For fixed k, this term is maximized if $m^2 - km + k^2/8 = 0$. Since $m \leq k/2$, only the smaller solution $m = (1/2 - 1/\sqrt{8})k$ is possible. Invoking this m into the above estimation we obtain after some simplifications: $\log E < 1.275k$; the tedious calculations are omitted. With k from Lemma 5.1 we get the assertion. □

Acknowledgment

I am grateful to the anonymous referees for their many valuable comments.

Note

1. Here and in the following, such denotations mean that some universal constant factor, being independent of both n and r, is hidden in the $O(\,)$. This should not be confused with the assumption that r is "constant" in any concrete problem instance.

References

Almuallim, H. & Dietterich, T.G. (1994). Learning Boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, *69*, 279–305.

Alon, N., Bruck, J., Naor, J., Naor, M., & Roth, R. (1992a). Construction of asymptotically good, low-rate error-correcting codes through pseudorandom graphs. *IEEE Transactions on Information Theory*, *38*, 509–516.

Alon, N., Goldreich, O., Håstad, J., & Peralta, R. (1992b). Simple constructions of almost k-wise independent random variables. *Random Structures and Algortihms*, *3*, 289–304; Ibid. (1993) *4*, 119–120.

Althöfer, I. & Triesch, E. (1993). Edge search in graphs and hypergraphs of bounded rank. *Discrete Mathematics*, *115*, 1–9.

Angluin, D. (1987). Queries and concept learning. *Machine Learning*, *2*, 319–342.

Balding, D. J. & Torney, D. C. (1995). A comparative survey of non-adaptive pooling designs. In *Genetic mapping and DNA sequencing (IMA volumes in mathematics and its applications)* (pp. 133–155). Springer.

Balding, D. J. & Torney, D. C. (1996). Optimal pooling designs with error detection. *Journal of Combinatorial Theory A*, *74*, 131–140.

Beimel, A., Geller, F., & Kushilevitz, E. (1998). The query complexity of finding local minima in the lattice. In *Proceedings of the 11th Conference on Computational Learning Theory (COLT)* (pp. 294–302). ACM Press.

Ben-David, S., Kushilevitz, E., & Mansour, Y. (1997). Online learning versus offline learning. *Machine Learning*, *29*, 45–63.

Blum, A. (1992). Learning Boolean functions in an infinite attribute space. *Machine Learning*, *9*, 373–386.

Blum, A., Hellerstein, L., & Littlestone, N. (1995). Learning in the presence of finitely or infinitely many irrelevant attributes. *Journal of Computer and System Sciences*, *50*, 32–40.

Bshouty, N. H. (1995). Exact learning Boolean functions via the monotone theory. *Information and Computation*, *123*, 146–153.

Bshouty, N. H. & Cleve, R. (1992). On the exact learning of formulas in parallel. In *Proceedings of the 33th IEEE Foundations of Computer Science (FOCS)* (pp. 513–522). IEEE Press.

Bshouty, N. H. & Hellerstein, L. (1996). Attribute-efficient learning in query and mistake-bound models. In *Proceedings of the 9th Conference on Computational Lerning Theory (COLT)* (pp. 235–243). ACM Press.

Clausen, M., Dress, A., Grabmeier, & J., Karpinski, M. (1991). On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theoretical Computer Science*, *84*, 151–164.

Colbourn, C. J. & Dinitz, J. H. (1996). *The CRC Handbook of Combinatorial Designs*. CRC Press.

Damaschke, P. (1994). A tight upper bound for group testing in graphs. *Discrete Applied Mathematics*, *48*, 101–109.

Damaschke, P. (1997). The algorithmic complexity of chemical threshold testing. In *Lecture Notes in Computer Science, Vol. 1203: Proceedings of the 3rd Italian Conference on Algorithms and Complexity (CIAC)* (pp. 205–216). Springer.

Damaschke, P. (1998a). A chip search problem on binary numbers. In *Lecture Notes in Computer Science, Vol. 1380: Proceedings of the 3rd Latin American Symposium on Theoretical Informatics (LATIN)* (pp. 11–22). Springer.

Damaschke, P. (1998b). Comutational aspects of parallel attribute-efficient learning. In *Lecture Notes in Artificial Intelligence, Vol. 1501: Proceedings of the 9th International Workshop on Algorithmic Learning Theory (ALT)* (pp. 103–111). Springer.

Damaschke, P. (1998c). Randomized group testing for mutually obscuring defectives. *Information Processing Letters*, *67*, 131–135.

De Bonis, A., Gargano, L., & Vaccaro, U. (1998). Improved algorithms for chemical threshold testing problems. In *Lecture Notes in Computer Science, Vol. 1449: Proceedings of the 4th Conference on Computing and Combinatorics (COCOON)* (pp. 127–136). Springer.

De Bonis, A. & Vaccaro, U. (1998). Improved algorithms for group testing with inhibitors. *Information Processing Letters*, *67*, 57–64.

Dhagat, A. & Hellerstein, L. (1994). PAC learning with irrelevant attributes. In *Proceedings of the 35th IEEE Foundations of Computer Science (FOCS)* (pp. 64–74). IEEE Press.

Du, D. Z. & Hwang, F. K. (1993). *Combinatorial Group Testing and its Applications*. World Scientific.

Farach, M., Kannan, S., Knill, E., & Muthukrishnan, S. (1997). Group testing problems in experimental molecular biology. In *Proceedings of Compression and Complexity of Sequences* (pp. 357–367). IEEE Computer Society.

Fischer, P., Klasner, N., & Wegener, I. (1999). On the cut-off point for combinatorial group testing. *Discrete Applied Mathematics*, *91*, 83–92.

Goldman, S. A. & Sloan, R. H. (1994). The power of self-directed learning. *Machine Learning*, *14*, 271–294.

Hofmeister, T. (1999). An application of codes to attribute-efficient learning. In *Lecture Notes in Artificial Intelligence, Vol. 1572: Proceedings of 5th European Conference on Computational Learning Theory* (EuroCOLT) (pp. 101–110). Springer.

Khardon, R. & Roth, D. (1996). Reasoning with models. *Artificial Intelligence*, *87*, 187–213.

Kivinen, J., Mannila, H., & Ukkonen, E. (1992). Learning hierarchical rule sets. In *Proceedings of the 5th Conference on Computational Learning Theory (COLT)* (pp. 37–44). ACM Press.

Kleitman, D. J. & Spencer, J. H. (1973). Families of k-independent sets. *Discrete Mathematics*, *6*, 255–262.

Knill, E. (1995). Lower bounds for identifying subset members with subset queries. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms (SODA)* (pp. 369–377).

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Macula, A. & Reuter, G. Simplified searching for two defects. *Journal of Statistical Planning and Inference*. To appear.

Motwani, R. & Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press.

Naor, J. & Naor, M. (1993). Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, *22*, 838–856.

Naor, M., Schulman, L. J., & Srinivasan, A. (1995). Splitters and near-optimal derandomization. In *Proceedings of the 36th IEEE Foundations of Computer Science (FOCS)* (pp. 182–191). IEEE Press.

Roth, R. M. & Benedek, G. M. (1991). Interpolation and approximation of sparse multivariate polynomials over GF(2). *SIAM Journal on Computing*, *20*, 291–314.

Seroussi, G. & Bshouty, N. H. (1988). Vector sets for exhaustive testing of logic circuits. *IEEE Transactions on Information Theory*, *34*, 513–522.

Triesch, E. (1996). A group testing problem for hypergraphs of bounded rank. *Discrete Applied Mathematics*, *66*, 185–188.

Uehara, R., Tsuchida, K., & Wegener, I. (1997). Optimal attribute-efficient learning of disjunction, parity, and threshold functions. In *Lecture Notes in Artificial Intelligence, Vol. 1208: Proceedings of the 3rd European Conference on Computational Learning Theory (EuroCOLT)* (pp. 171–184). Springer.

Received March 24, 1999
Revised November 17, 1999
Final manuscript September 21, 1999