Electronic Colloquium on Computational Complexity, Report No. 66 (2006)

On Attribute Efficient and Non-adaptive Learning of Parities and DNF Expressions*

Vitaly Feldman†
Harvard University
Cambridge, MA 02138
VITALY@EECS.HARVARD.EDU

Editor:

Abstract

We consider the problems of attribute-efficient PAC learning of two well-studied concept classes: parity functions and DNF expressions over $\{0,1\}^n$. We show that attribute-efficient learning of parities with respect to the uniform distribution is equivalent to decoding high-rate random linear codes from low number of errors, a long-standing open problem in coding theory.

An algorithm is said to use membership queries (MQs) *non-adaptively* if the points at which the algorithm asks MQs do not depend on the target concept. Using a simple non-adaptive parity learning algorithm and a modification of Levin's algorithm for locating a weakly-correlated parity due to Bshouty *et al.,* we give the first non-adaptive and attribute-efficient algorithm for learning DNF with respect to the uniform distribution. Our algorithm runs in time $\tilde{O}(ns^4/\epsilon)$ and uses $\tilde{O}(s^4/\epsilon)$ non-adaptive MQs where s is the number of terms in the shortest DNF representation of the target concept. The algorithm improves on the best previous algorithm for learning DNF (of Bshouty *et al.*) and can also be easily modified to tolerate random classification noise in MQs.

Keywords: Attribute-efficient, parity, non-adaptive membership query, DNF

1. Introduction

The problems of PAC learning parity functions and DNF expressions are among the most fundamental and well-studied problems in machine learning theory. Along with running time efficiency, an important consideration in the design of learning algorithms is their *attribute efficiency.* A class $\mathcal{C}$ of Boolean functions is said to be *attribute-efficiently learnable* if there is an efficient algorithm which can learn any function $f \in \mathcal{C}$ using a number of examples which is polynomial in the "size" (description length) of the function f to be learned, rather than in n, the number of attributes in the domain over which learning takes place. Attribute-efficiency arises naturally from a ubiquitous practical scenario in which the total number of potentially influential attributes is much larger than the number of relevant attributes (i.e., the attributes on which the concept actually depends), whereas examples are either scarce or expensive to get.

∗. Parts of this work are published in the Proceedings of 18th Annual Conference on Learning Theory, 2005
†. Supported by grants from the National Science Foundation NSF-CCF-9877049, NSF-CCF-0432037, and NSF-CCF-0427129.

ISSN 1433-8092

Learning of DNF expressions and attribute-efficient learning of parities from random examples with respect to the uniform distribution are both long-standing challenges in learning theory. Lack of substantial progress on these questions has resulted in attempts to solve them in stronger learning models. The most well-studied such model is one in which a *membership query oracle* is given to the learner in addition to the example oracle. The learning algorithm may query this oracle for a value of the target function at any point of its choice. Jackson gave the first algorithm that learns DNF from membership queries (MQs) under the uniform distribution [Jac94] and later Bshouty, Jackson and Tamon gave a more efficient and attribute-efficient algorithm for learning DNF in the same setting [BJT99]. The first algorithm for attribute-efficient learning of parities using MQs is due to Blum, Hellerstein and Littlestone [BHL95], and their result was later refined by Uehara *et al.* [UTW97].

A number of later works gave learning algorithms for DNF expressions in models where the learning algorithm is more passive than in the MQ model [JSS97, BF02, BMOS03]. A restricted model of membership queries, which addresses some of the disadvantages of the MQ model, is the model in which MQs are asked non-adaptively. An algorithm is said to use MQs *non-adaptively* (in our context we will often call it non-adaptive for brevity) if the queries of the algorithm do not depend on the target concept. In other words, the learning algorithm can be split into two stages. The first stage, given the learning parameters, generates a set S of queries for the membership oracle. The second one, given the answers to the queries in S, produces a hypothesis (without further access to the oracle). An immediate advantage of this model (over the usual MQ) is the fact that the queries to the membership oracle can be parallelized. This, for example, is crucial in DNA sequencing and other biological applications where tests are very time-consuming but can be parallelized (*cf.* [FKKM97, Dam98] and references therein). Another advantage of a non-adaptive learner is that the same set of points can be used to learn numerous concepts. This seems to be happening in human brain where a single example can be used in learning of several different concepts and hence systems that aim to reproduce learning abilities of the human brain need to possess this property [Val94, Val00, Val05]. It is important to note that in the two practical applications mentioned above, attribute-efficiency is also a major concern. It is therefore natural to ask: which classes can be PAC learned attribute-efficiently by non-adaptive MQs? We refer to this model of learning as *ae.naMQ learning.* This question was first explicitly addressed by Damaschke [Dam98] who proved that any function of r variables is ae.naMQ learnable when it is represented by the truth table of the function (requiring $r \log n + 2^r$ bits). Later Hofmeister gave the first ae.naMQ algorithm for learning parities [Hof99] and Guijarro *et al.* gave an algorithm for learning functions of at most $\log n$ variables in the decision tree representation [GLR99]. But the question remains open for numerous other representations used in learning theory.

1.1 Our Results

We first establish the equivalence between attribute-efficient learning of parities from random uniform examples [BHL95] and decoding high-rate random linear codes from low number of errors a long-standing open problem in coding theory widely believed intractable. The latter is equivalent to learning of parities with adversarial classification noise. Thus we may consider this equivalence as a new evidence of the hardness of attribute-efficient learning of parities from random examples only. This result together with a recent result of Feldman *et al.* [FGKP06] implies equivalence of attribute-efficient learning of parities and learning of parities with random noise (for an appropriate transformation of parameters).

We give a simple and fast randomized algorithm for ae.naMQ learning of parities and show transformation that converts a non-adaptive parity learning algorithm into an algorithm for finding heavy Fourier coefficients of a function while preserving attribute-efficiency and non-adaptiveness. Using these components we give the first ae.naMQ algorithm for learning DNF expressions with respect to the uniform distribution. It runs in time $\tilde{O}(ns^4/\epsilon)$ and uses $\tilde{O}(s^4 \log^2 n/\epsilon)$ MQs (where s is the DNF-size of the target concept). The algorithm improves on the $\tilde{O}(ns^6/\epsilon^2)$-time and $\tilde{O}(ns^4 \log n/\epsilon^2)$-query algorithm of Bshouty *et al.* We also show a simple and general modification that allows the above algorithm to efficiently handle random persistent classification noise in MQs.

1.2 Previous Results

Blum *et al.* were the first to ask whether parities are learnable attribute-efficiently (in the related *on-line mistake-bound* model) [BHL95]. They also presented the first algorithm to learn parity functions attribute-efficiently using MQs. Their algorithm is based on the following approach. First all the relevant attributes are identified and then a simple (not attribute-efficient) algorithm restricted to the relevant variables is used to learn the concept. Since then other algorithms were proposed for attribute-efficient identification of relevant variables [BH98, GTT99]. All the algorithms are based on a binary search for a relevant variable given a positive and a negative example. Binary search and the fact that queries in the second stage depend on the variables identified in the first stage only allows for the construction of adaptive algorithms via this approach. Uehara *et al.* gave several algorithms for attribute-efficient learning of parities that again used adaptiveness in an essential way [UTW97].

Equivalence of attribute-efficient learning of parities by membership queries and linear codes was earlier observed by Hofmeister [Hof99]. He also gave the first ae.naMQ algorithm for learning parities based on BCH codes. Our (independently-obtained) result can be seen as an extension of this equivalence to random uniform examples.

Little previous work has been published on attribute-efficient learning of parities from random examples. Indeed, the first non-trivial result in this direction has only recently been given by Klivans and Servedio [KS04]. They prove that parity functions on at most k variables are learnable in polynomial time using $O(n^{1-\frac{1}{k}} \log n)$ examples.

Efficient learning of unrestricted DNF formulas under the uniform distribution begins with a famous result by Jackson [Jac94]. The algorithm, while polynomial-time, is somewhat impractical due to the $\tilde{O}(ns^{10}/\epsilon^{12})$ bound on running time. By substantially improving the key components of Jackson's algorithm the works of Freund [Fre92], Bshouty *et al.* [BJT99], and Klivans and Servedio [KS03] resulted in an algorithm that learns DNF in time $\tilde{O}(ns^6/\epsilon^2)$ and uses $\tilde{O}(ns^4/\epsilon^2)$ MQs[1]. This algorithm is non-adaptive but also not attribute-efficient. Using the algorithm for identification of relevant variables by Bshouty

1. Bshouty *et al.* claimed sample complexity $\tilde{O}(ns^2/\epsilon^2)$ but this was in error as explained in Remark 13.

and Hellerstein mentioned above Bshouty *et al.* gave an attribute-efficient version of their algorithm running in time $\tilde{O}(rs^6/\epsilon^2 + n/\epsilon)$ and using $\tilde{O}(rs^4 \log n/\epsilon^2)$ adaptive MQs.

2. Preliminaries

General. For vectors $x, y \in \{0,1\}^n$ we denote by $x|y$ the vector obtained by concatenating x with y; by $x \oplus y$ the vector obtained by bitwise XOR of x and y; by $[k]$ the set $\{1, 2, \ldots, k\}$; by e_i a vector with 1 in i-th position and zeros in the rest; by x_i the i-th element of vector x; by M_i the i-th column of matrix M; and define $x_{[i,j]} = x_i|x_{i+1}|\cdots|x_j$. Dot product $x \cdot y$ of vectors $x, y \in \{0,1\}^n$ denotes $\sum_i x_i y_i \pmod 2$ or simply vector product xy^T over $\mathbf{GF}(2)$ (with vectors being row vectors by default). By $\mathtt{wt}(x)$ we denote the Hamming weight of x and we define $\mathtt{dist}(x, y) = \mathtt{wt}(x \oplus y)$.

To analyze accuracy and confidence of estimates produced by random sampling besides the more standard Chernoff and Hoeffding bounds, we use Bienaymé-Chebyshev's inequality for pairwise independent samples.

Lemma 1 (Bienaymé-Chebyshev) *Let $X_1, \ldots, X_m$ be pairwise independent random variables all with mean μ and variance σ^2. Then for any $\lambda \geq 0$,*

$$\mathbf{Pr}\left[\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| \geq \lambda\right] \leq \frac{\sigma^2}{m\lambda^2} \ .$$

We study learning of Boolean functions on the Boolean cube $\{0,1\}^n$. Our Boolean functions take values $+1$ (true) and -1 (false). Our main interest are the classes of parity functions and DNF expressions. Parity function $\chi_a(x)$ for a vector $a \in \{0,1\}^n$ is defined as $\chi_a(x) = (-1)^{a \cdot x}$. We refer to the vector associated with a parity function as its *index*. We denote the concept class of parity functions $\{\chi_a \mid a \in \{0,1\}^n\}$ by PAR and the class of all the parities on at most k variables by PAR(k). We represent a parity function by listing all the variables on which it depends. This representation for a parity on k variables requires $\theta(k \log n)$ bits.

For the standard DNF representation and any Boolean function f we denote by DNF-size(f) the number of terms in a DNF representation of f with the minimal number of terms. In context of learning DNF this parameter is always denoted s. The uniform distribution over $\{0,1\}^n$ is denoted $\mathcal{U}$.

PAC Learning. Our learning model is Valiant's well-known PAC model [Val84] for learning Boolean functions over $\{0,1\}^n$. In this model, for a concept c and distribution $\mathcal{D}$ over X, an *example oracle* $\mathrm{EX}_{\mathcal{D}}(c)$ is an oracle that upon request returns an example $\langle x, c(x)\rangle$ where x is chosen randomly with respect to $\mathcal{D}$, independently of any previous examples. For $\epsilon \geq 0$ we say that function g ϵ-approximates a function f with respect to distribution $\mathcal{D}$ if $\mathbf{Pr}_{\mathcal{D}}[f(x) = g(x)] \geq 1 - \epsilon$. We say that an algorithm $\mathcal{A}$ (efficiently) learns concept class $\mathcal{C}$ if for every $\epsilon > 0$, n, $c \in \mathcal{C}$, and distribution $\mathcal{D}$ over $\{0,1\}^n$, $\mathcal{A}(n, \epsilon, s)$ (where s is the size of c in the representation associated with $\mathcal{C}$) outputs, with probability at least 1/2, an efficiently computable hypothesis h that ϵ-approximates c. When a learning algorithm is guaranteed to learn only with respect to a specific distribution we specify the distribution explicitly.

Membership query oracle MEM(c) is the oracle that, given any point $x \in \{0,1\}^n$, returns the value $c(x)$. When learning with respect to $\mathcal{U}$, $\mathrm{EX}_{\mathcal{U}}(c)$ can be trivially simulated using MEM(c) and therefore $\mathrm{EX}_{\mathcal{U}}(c)$ is not used at all.

An algorithm $\mathcal{A}$ is said to be attribute-efficient if the number of examples (both random and received from MQ oracle) it uses is polynomial in the size of the representation of the concept. We say that a variable x_i is *relevant* for a function f if there exists $y \in \{0,1\}^n$ such that $f(y) \neq f(y \oplus e_i)$. The number of relevant variables of the target concept is denoted by parameter r.

For a function $t(\cdots)$ we say a function $q(\cdots)$ (of the same parameters as t) is $\tilde{O}(t(\cdots))$ when there exist constants α and β such that $q(\cdots) \leq \alpha t(\cdots) \log^{\beta}(t(\cdots))$.

Learning by Non-adaptive Membership Queries. We say that an algorithm $\mathcal{A}$ uses MQs *non-adaptively* if it can be split into two stages. The first stage, given all the parameters of learning, (n, ϵ and a bound on the size of the target concept), generates a set of points $S \subseteq \{0,1\}^n$. The second stage, given the answers from MEM(c) on points in S, i.e. the set $\{(x, c(x)) \mid x \in S\}$, computes a hypothesis (or, in general, performs some computation). Neither of the stages has any other access to MEM(c). We note that in the general definition of PAC learning we did not assume that size of the target concept (or a bound on it) is given to the learning algorithm. When learning with adaptive queries a good bound can be found via the "guess-and-double" technique but for adaptive algorithms we will assume that this bound is always given. Clearly the same "guess-and-double" technique can be used to produce a sequence of independent and non-adaptive executions of the learning algorithm.

The immediate consequence of non-adaptiveness is that in order to parallelize a non-adaptive learning algorithm only the usual computation has to be parallelized since all the MQs can be made in parallel. Another simple consequence is for parallel learning of ℓ concepts from the same concept class. The fact that queries are independent of the target concept implies that same set of points can be used for learning different concepts. To achieve probability of success $1/2$ in learning of all ℓ concepts we will have to learn with each concept with probability of success $1 - 1/(2\ell)$. This implies that the number of points needed for learning might grow by a factor of $\log \ell$ whereas in the general case ℓ times more examples might be required.

Valiant has proved that if one-way functions exist then there exists a concept class not learnable even with access to MQ oracle and only with respect to $\mathcal{U}$ [Val84]. A simple modification of his proof can be used to show that if one-way functions exist then use of non-adaptive MQ is strictly weaker than (adaptive) MQs and is strictly stronger than use random examples only.

Fourier transform. The Fourier transform is a technique for learning with respect to the uniform distribution (primarily) based on the fact that any function f over $\{0,1\}^n$ can be represented as a linear combination of parities, that is $f(x) = \sum_{a \in \{0,1\}^n} \hat{f}(a)\chi_a(x)$. The coefficient $\hat{f}(a)$ is called Fourier coefficient of f on a and equals $\mathbf{E}_{\mathcal{U}}[f(x)\chi_a(x)]$; a is called the *index* and $\mathtt{wt}(a)$ the *degree* of $\hat{f}(a)$. All Fourier coefficients (or the Fourier transform) can be computed via the Fast Fourier Transform algorithm in time $O(n2^n)$. The same transformation also converts Fourier coefficients into the values of the function f on all the points and is called inverse Fourier transform [CT65]. For further details on the technique we refer the reader to the survey by Mansour [Man94].

Boolean linear codes. We say that a code C is an $[m, n]$ code if C is a binary linear code of block length m, message length n. Any such code can be described by its $n \times m$ *generator matrix* G as follows: $C = \{xG \mid x \in \{0,1\}^n\}$. Equivalently, a code can be described by its *parity-check* matrix H of size $m \times (m-n)$ by $C = \{y \mid yH = \bar{0}\}$. It is well-known (and easy to see) that $G \cdot H = 0^{n \times (m-n)}$ and decoding given a corrupted message y is equivalent to decoding given the *syndrome* of the corrupted message. The syndrome equals to yH and the decoding consists of finding a vector e of Hamming weight at most d such that $y \oplus e = xG$, where d is a bound on the number of errors the code can correct.

By saying that C is a *random* $[m, n]$ *code* we mean that C is defined by choosing randomly, uniformly, and independently n vectors in $\{0,1\}^m$ that form the basis of C. Alternatively, we can say that the generator matrix G of C was chosen randomly with each entry equal to 1 with probability 1/2 independently of others. We denote this distribution by $\mathcal{U}_{n \times m}$.

3. Attribute-Efficient Learning of Parities

In this section we show that attribute-efficient learning of parities from uniform random examples only is likely to be hard by proving that it is equivalent to an open problem in coding theory. We also give a simple and fast algorithm for ae.naMQ learning of parities. Unlike in the rest of the paper in this section parity functions will be $0,1$ functions. To emphasize this we use $\dot{\chi}$ instead of χ.

3.1 Learning of Parities and Binary Linear Codes

The equivalence of attribute-efficient learning of parities with respect to the uniform distribution and decoding of random linear codes relies on two observations. The first one, due to Hofmeister [Hof99], is that attribute-efficient learning of parities is exactly the syndrome decoding of a linear code. The second one is that an equivalent way to generate a random linear code is to generate a random parity check matrix (instead of the random generator matrix).

The first observation follows immediately from the definition of syndrome decoding since $yH = eH$ and $\mathtt{wt}(e) \le d$ (see Section 2). Therefore yH equals to evaluation of parity χ_e on the columns of H. Hence syndrome decoding of d-error correcting code is the same as learning of PAR(d) from evaluations on columns of H. Let $\mathcal{V}_{n \times m}$ denote the distribution on matrices of size $n \times m$ resulting from the following process. Choose a random matrix H of size $m \times (m-n)$ of rank $m-n$ and then choose randomly and uniformly a matrix G of size $n \times m$ of rank n such that $G \cdot H = 0^{n \times (m-n)}$. Let $p(i,j)$ denote the probability that i vectors chosen randomly and uniformly from $\{0,1\}^j$ are linearly independent. By definition, $p(i,j) = (1-2^{-j}) \cdot (1-2^{-j+1}) \cdots (1-2^{-j+i-1})$. We first give the following simple lemma on the distribution $\mathcal{V}_{n \times m}$.

Lemma 2 *$\mathcal{V}_{n \times m}$ is efficiently samplable.*

Proof First note that for $x < 1$, $(1 - \frac{x}{2}) \ge (1 - x + \frac{x^2}{2}) \ge e^{-x}$. Therefore

$$p(i,j) \ge e^{-2^{-j+1}} e^{-2^{-j+2}} \cdots e^{-2^{-j+i}} = e^{-2^{-j+i+1}+2^{-j+1}} > e^{-2^{-j+i+1}} \ . \qquad (1)$$

This means that for any $i < j$, $p(i,j) > e^{-2}$. Therefore a randomly and uniformly chosen matrix H of size $m \times (m-n)$ will have rank $m-n$ with probability $p(m-n,m)$ which is at least a constant. We can then find a basis $b_1, \ldots, b_m$ for the subspace of $\{0,1\}^m$ that is "orthogonal" to H in the standard (and efficient) way. Let G_0 denote the matrix whose rows are the vectors $b_1, \ldots, b_m$. It is easy to see that any matrix G of rank m such that $GH = 0^{n\times(m-n)}$, can be represented uniquely as $F \cdot G_0$ where F is a matrix of size $n \times n$ and full rank. Therefore we can generate G's as above by choosing randomly and uniformly a matrix F of rank n. If we choose a random matrix F according $\mathcal{U}_{n\times n}$, with probability at least $p(n,n)$, it will have the full rank. Altogether, we can generate a matrix according to $\mathcal{V}_{n\times m}$ with probability at least some constant $c > 0$ in time $O(n^3)$ (or less if a non-trivial matrix multiplication algorithm is used). ■

The second observation is based on the following lemma.

Lemma 3 *The statistical distance between $\mathcal{V}_{n\times m}$ and $\mathcal{U}_{n\times m}$ is at most 2^{-m+n+2}.*

Proof Let G be any matrix of size $n \times m$ with linearly independent rows. Its probability under $\mathcal{U}_{n\times m}$ is $\mathcal{U}_{n\times m}(G) = 2^{-mn}$. When sampling with respect to $\mathcal{V}_{n\times m}$, G can be obtained only if all the columns of H are "orthogonal" to rows of G, that is belong to a linear subspace of $\{0,1\}^m$ of dimension $m-n$. Total number of H's like these of rank $m-n$ is $2^{(m-n)^2}p(m-n, m-n)$ and the total number of matrices size $m \times (m-n)$ of rank $m-n$ is $2^{m(m-n)}p(m-n, m)$. Therefore the probability of getting each H like this is $2^{-n(m-n)}\frac{p(m-n,m-n)}{p(m-n,m)}$. Given H the total number of matrices of size $n \times m$ and rank n that are orthogonal to H is $p(n,n)2^{n^2}$ and therefore G will be generated with probability $2^{-n^2}/p(n,n)$. Hence the total probability of G under $\mathcal{V}_{n\times m}$ is $\mathcal{V}_{n\times m}(G) = 2^{-mn}\frac{p(m-n,m-n)}{p(m-n,m)p(n,n)}$. For every $i < j$, $p(j-i,j)p(i,i) = p(j,j)$. Therefore $\mathcal{V}_{n\times m}(G) = 2^{-mn}/p(n,m)$. This implies that the statistical distance between $\mathcal{V}_{n\times m}$ and $\mathcal{U}_{n\times m}$ is at most $2(1-p(n,m))$. According to equation (1), $2(1-p(n,m)) \leq 2(1-e^{-2^{-m+n+1}}) \leq 2^{-m+n+2}$. ■

We can now present the exact statement of equivalence between attribute-efficient learning and decoding of random linear codes.

Theorem 4 *Assume that there exists an algorithm* `AELearnPar` *that efficiently learns $PAR(k)$ over $\{0,1\}^m$ using at most $q(k, \log m)$ random examples. Then there exists an algorithm* `RandDec`*, that for a randomly chosen $[m,n]$ code C, where $n = m - q(k, \log m)$, and any $y \in \{0,1\}^m$ such that $\exists x \in \{0,1\}^n$, $\mathtt{dist}(C(x), y) \leq k$, runs in polynomial time and, with probability at least $1/2 - o(1)$, (over the choice of C and the random choices of* `RandDec`*) finds x.*

Proof Let G and $y = xG \oplus e$ such that $\mathtt{wt}(e) \leq k$ be the input to `RandDec`. If G is not of rank n we just return the vector 0^n. Otherwise we use G to generate a random matrix H according to distribution $\mathcal{V}_{m-n,n}$ (as in Lemma 2). The syndrome of y, yH is equal to eH which is equal to the vector $\dot{\chi}_e(H_1), \dot{\chi}_e(H_2), \ldots, \dot{\chi}_e H_{m-n}$ where H_i is the i-th column of H. The statistical distance between the H generated as above and a randomly and uniformly chosen matrix is at most $2^{-m+(m-n)+2} = 2^{-n+2}$. Therefore with probability at least $1/2 - 2^{-n+2}$, `AELearnPar` will return e when given the examples $\langle H_1, yH_1\rangle, \langle H_2, yH_2\rangle, \ldots, \langle H_q, yH_q\rangle$. Given e we can easily find x. This algorithm succeeds with probability at least $1/2 - (1-p(n,m)) - 2^{-n+2} = 1/2 - o(1)$ for superconstant n and q. ■

It is easy to see from the proof of Theorem 4 that the converse is also true, giving us the following theorem.

Theorem 5 *Assume that there exists an algorithm* `RandDec` *that for a randomly chosen* $[m, n]$ *code* C *and any* $y \in \{0,1\}^m$ *such that* $\exists x \in \{0,1\}^n$, $\texttt{dist}(C(x), y) \leq d$, *runs in polynomial time and, with probability at least* $1/2 + o(1)$ *(over the choice of* C *and the random choices of* `RandDec`*), finds* x. *Then PAR*(d) *over* $\{0,1\}^m$ *is efficiently learnable from* $m - n$ *random examples.*

Another famous open problem in learning theory, learning of parities with random classification noise is equivalent to decoding of random linear codes with random errors that is likely to be easier than decoding with adversarial errors. However Feldman *et al.* recently showed that decoding with adversarial noise of rate η can be reduced to decoding with random noise of rate $2\eta - 2\eta^2$ [FGKP06]. This allows us to relate the two learning problems directly, in particular, we obtain the following theorem.

Theorem 6 *Assume that there exists an algorithm* `LearnPar` *that efficiently learns PAR over* $\{0,1\}^n$ *with random noise of rate* η *using at most* $q(n, \eta)$ *random examples. Then there exists an efficient algorithm* `AELearnPar` *that learns PAR*$(\frac{\eta \cdot m}{2})$ *over* $\{0,1\}^m$ *using at most* $m - n$ *examples, where* $m = q(n, \eta)$.

Equivalence in the other direction is the same as Theorem 4 since decoding of $[m, n]$ code from k errors is the same as learning of parities over $\{0,1\}^n$ with random noise of rate $\eta = k/n$ (in fact, even adversarial) from m examples.

3.2 A Fast Randomized Algorithm for ae.naMQ Learning of Parities

We next present a simple randomized algorithm for ae.naMQ learning of parities. Previous ae.naMQ algorithms for learning parities by Hofmeister relied on decoding of specific binary linear codes (as BCH or Reed-Solomon) and require superlinear in n time [Hof99].

Theorem 7 *For each* $k \leq n$ *there exists an algorithm that ae.naMQ learns the class PAR*(k) *in time* $O(nk \log(n/\delta))$ *and asks* $O(k \log(n/\delta))$ *MQs.*

Proof Let $\dot{\chi}_c$ be the target concept (such that $\texttt{wt}(c) \leq k$). We define $\mathcal{D}_{\frac{1}{t}}$ to be the product distribution such that for each i, $\mathbf{Pr}[x_i = 1] = \frac{1}{t}$. Let us draw a point x randomly according to distribution $\mathcal{D}_{\frac{1}{4k}}$. Then for each $i \leq n$

$$\begin{aligned}\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[x_i = 1 \text{ and } \dot{\chi}_c(x) = 1] &= \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1]\ \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[x_i = 1] \\ &= \frac{1}{4k}\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1]\ .\end{aligned}$$

Our second observation is that for any set of indices $B \subseteq [n]$ and the corresponding parity function $\dot{\chi}_b$,

$$\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_b(x) = 1] \leq 1 - \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\forall i \in B,\ x_i = 0] = 1 - (1 - \frac{1}{4k})^{|B|} \leq \frac{|B|}{4k}\ .$$

We now assume that $c_i \neq 1$ and therefore does not influence $\dot{\chi}_c$. Then by the second observation

$$\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1] = \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1] \leq \frac{k}{4k} \leq 1/4 \ .$$

Now assume that $c_i = 1$ and let $c' = c \oplus e_i$. Then $\dot{\chi}_c(x) = 1$ if and only if $\dot{\chi}_{c'}(x) = 0$ and $\dot{\chi}_{c'}(x)$ is independent of x_i. Therefore

$$\begin{aligned}\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1] &= \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_{c'}(x) = 0 \mid x_i = 1] \\ &= 1 - \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_{c'}(x) = 1] \geq 1 - \frac{k-1}{4k} > 3/4 \ .\end{aligned}$$

Hence estimation of $\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[x_i = 1 \text{ and } \dot{\chi}_c(x) = 1]$ within the half of the expectation can be used to find out whether $c_i = 1$. By taking $\alpha k \log(n/\delta)$ independent samples[2] with respect to $\mathcal{D}_{\frac{1}{4k}}$ (for some constant $\alpha \geq 32 \ln 2$) we will get that each estimate is correct with probability at least $1 - \delta/n$ and therefore we will discover c with probability at least $1 - \delta$. The running time of resulting algorithm is clearly $O(nk \log(n/\delta))$. ■

4. Weak Parity Learning

The original Jackson's algorithm for learning DNF expressions with respect to the uniform distribution is based on a procedure that weakly learns DNF with respect to the uniform distribution [Jac94]. The procedure for weak learning is essentially an algorithm that, given a Boolean function f finds one of its heavy Fourier coefficients, if one exist. Jackson's algorithm is based on a technique by Goldreich and Levin for finding a heavy Fourier coefficient [GL89]. Bshouty, Jackson, and Tamon used a later algorithm by Levin [Lev93] to give a significantly faster weak learning algorithm [BJT99]. Below we briefly describe Levin's algorithm with improvements by Bshouty *et al.* Detailed proofs of all the statements and smaller remarks can be found in the paper by Bshouty *et al.* [BJT99](Sect. 4) (we follow their definitions and notation to simplify the reference).

A Fourier coefficient $\hat{f}(a)$ of a function $f : \{0,1\}^n \to \{-1,+1\}$ is said to be θ-heavy if $|\hat{f}(a)| \geq \theta$.

Definition 8 (Weak Parity Learning) *Given $\theta > 0$ and access to MEM(f) for a Boolean function f that has at least one θ-heavy Fourier coefficient the weak parity learning problem consists of finding the index a $\theta/2$-heavy Fourier coefficient of f.*

We will only consider algorithms for weak parity learning that are efficient, that is, produce the result with probability at least 1/2 in time polynomial in n, and θ^{-1}. In addition we are interested in weak parity learning algorithms that are attribute-efficient.

Definition 9 (Attribute-Efficient Weak Parity Algorithm) Attribute-efficient weak parity algorithm *is an algorithm that given k, θ, and MEM(f) for f that has a θ-heavy*

2. It is important to use the multiplicative and not the additive form of Chernoff bounds to get linear dependence on k.

Fourier coefficient of degree at most k efficiently solves weak parity learning problem and asks polynomial in $k, \log n$, and θ^{-1} number of MQs.

Attribute-efficient weak learning of DNF can be obtained from an attribute-efficient weak parity algorithm via the following lemma by Bshouty and Feldman.

Lemma 10 ([BF02](Lemma 18)) *For any Boolean function f of DNF-size s and a distribution $\mathcal{D}$ over $\{0,1\}^n$ there exists a parity function χ_a such that*

$$|\mathbf{E}_{\mathcal{D}}[f\chi_a]| \geq \frac{1}{2s+1} \text{ and } \mathtt{wt}(a) \leq \log\left((2s+1)L_\infty(2^n\mathcal{D})\right) .$$

Levin's algorithm is based on estimating a Fourier coefficient $\hat{f}(a)$ by sampling f on randomly-chosen pairwise independent points. More specifically, the following pairwise independent distribution is generated. For a fixed k, a random m-by-n 0-1 matrix R is chosen and the set $Y = \{pR \mid p \in \{0,1\}^m - \{0^m\}\}$ is formed. Bienaymé-Chebyshev's inequality implies that

$$\mathbf{Pr}_R\left[\left|\frac{\sum_{x\in Y} f(x)\chi_a(x)}{2^m-1} - \hat{f}(a)\right| \geq \gamma\right] \leq \frac{1}{(2^m-1)\gamma^2} \quad (2)$$

Therefore using a sample for $m = \log(16\rho^{-1}\theta^{-2}+1)$, $\sum_{x\in Y} f(x)\chi_a(x)$ will, with probability at least $1-\rho$, approximate $\hat{f}(a)$ within $\theta/4$.

On the other hand, $\sum_{x\in Y} f(x)\chi_a(x)$ is a summation over all (but one[3]) elements of a linear subspace of $\{0,1\}^n$ and therefore can be seen as a Fourier coefficient of f restricted to the subspace Y. That is, if we define $f_R(p) = f(pR)$ then, by definition of Fourier transform, for every $z \in \{0,1\}^m$

$$\widehat{f_R}(z) = 2^{-m}\sum_{p\in\{0,1\}^m} f_R(p)\chi_z(p) .$$

This together with equality $\chi_a(pR) = \chi_{aR^T}(p)$ implies that $\hat{f}(a)$ is approximated by $\widehat{f_R}(aR^T)$ (with probability at least $1-\rho$).

All the coefficients $\widehat{f_R}(z)$ can be computed exactly in time $|Y|\log|Y|$ via the FFT algorithm giving estimations to all the Fourier coefficients of f.

Another key element of the weak parity algorithm is the following equation. For $c \in \{0,1\}^n$ let $f_c(x) = f(x\oplus c)$. Then

$$\widehat{f_c}(a) = 2^{-n}\sum_{x\in\{0,1\}^n} f(x\oplus c)\chi_a(x) = 2^{-n}\sum_{x\in\{0,1\}^n} f(x)\chi_a(x\oplus c) = \hat{f}(a)\chi_a(c) . \quad (3)$$

Assuming that $\hat{f}(a) \geq \theta$ estimation of $\hat{f}(a)$ within $\theta/4$ (when successful) has the same sign as $\hat{f}(a)$. Similarly we can obtain the sign of $\widehat{f_c}(a)$. The sign of the product $\hat{f}(a)\widehat{f_c}(a)$ is equal to $\chi_a(c)$. This gives a way to make MQs for χ_a using the values $\widehat{f_{c,R}}(aR^T)$ for a random R and leads to the following result.

3. The value at 0^m does not influence the estimation substantially and therefore can be offset by slightly increasing the size of sample space Y [BJT99].

Theorem 11 *Let $\mathcal{B}(k)$ be an ae.naMQ algorithm for learning parities that runs in time $t(n,k)$ and uses $q(\log n, k)$ MQs. There exists an attribute-efficient and non-adaptive weak parity learning algorithm* `WeakDNF`$-\mathcal{U}(\theta,\mathtt{k})$ *that runs in time $\tilde{O}\left(\theta^{-2}\cdot t(n,k)\cdot q(\log n,k)\right)$ and asks $\tilde{O}\left(\theta^{-2}\cdot q^2(\log n,k)\right)$ MQs.*

Proof Let $\mathcal{B}^2(k)$ be the algorithm $\mathcal{B}(k)$ repeated twice to get the probability of success to $3/4$ and let S be the set of MQs for an execution of $\mathcal{B}^2(k)$. Choose randomly an m-by-n matrix R for $m = \log\left(16\theta^{-2}\cdot 4\cdot(q(\log n,k)+1)+1\right)$ and compute the Fourier transforms of f_R and $f_{y,R}$ for each $y \in S$.

Then, for each $z \in \{0,1\}^m$ such that $|\widehat{f_R}(z)| \geq 3\theta/4$, we run $\mathcal{B}^2(k)$ with the answer to MQ $y \in S$ equal to $\mathtt{sign}(\widehat{f_R}(z)\widehat{f_{y,R}}(z))$ (here the non-adaptiveness of parity learning algorithm is essential). If the output of $\mathcal{B}^2(k)$ is a parity function on at most k variables we add it to the set of hypotheses H.

By Lemma 1, for a such that $|\hat{f}(a)| \geq \theta$ and $\mathtt{wt}(a) \leq k$, with probability at least $1 - \frac{1}{4(q(\log n,k)+1)}$, each of the estimations $\widehat{f_{y,R}}(aR^T)$ for $y \in S \cup \{0^k\}$ will be within $\theta/4$ of $\widehat{f_y}(a)$. In particular, all of them will have the right sign with probability at least $3/4$. When all the signs used for $\mathcal{B}^2$'s MQs are correct, $\mathcal{B}^2(k)$ succeeds with probability at least $3/4$. Therefore a will pass the magnitude test and will be added as a possible hypothesis with probability at least $1/2$. On the other hand for any $\hat{f}(b) < \theta/2$ it will not pass the magnitude test with probability at least $1 - \frac{1}{4(q(\log n,k)+1)} \geq 3/4$. Therefore by repeating this algorithm $O(1)$ times and choosing a vector a that appears in at least $3/8$ of all lists of hypotheses, we will find a $\theta/2$-heavy coefficient with probability at least $1/2$. It can be easily verified that time and sample complexity of the algorithm are as stated and its MQs are non-adaptive. ■

Another way to see Theorem 11 is as a way to convert an ae.naMQ algorithm for learning parities to an attribute-efficient algorithm for learning parities with adversarial classification noise (with respect to $\mathcal{U}$) of rate arbitrarily close to $1/2$. This follows from the fact that a parity function χ_c corrupted by noise of rate η has Fourier coefficient on c of weight at least $1-2\eta$.

We can now use the randomized parity learning algorithm and Theorems 10, 11 to get an algorithm for weakly learning DNF with the following properties.

Theorem 12 *There exist an algorithm* `WeakDNF-`$\mathcal{U}$ *that for a Boolean function f of DNF-size s given n, s, and access to $MEM(f)$, with probability at least $1/2$, finds a $(\frac{1}{2}-\Omega(\frac{1}{s}))$-approximator to f with respect to $\mathcal{U}$. Furthermore,* `WeakDNF-`$\mathcal{U}$ *runs in time $\tilde{O}\left(ns^2\right)$ and asks $\tilde{O}\left(s^2\log^2 n\right)$ non-adaptive MQs.*

The previous weak learning algorithm by Bshouty *et al.* requires $\tilde{O}\left(ns^2\right)$ MQs and runs in time[4] $\tilde{O}\left(ns^2\right)$.

4. The running time bound is based on use of a membership query oracle, that given any two vectors $x, y \in \{0,1\}^n$, passed to it "by reference", returns $f(x \oplus y)$ in $O(1)$ time.

5. Learning DNF Expressions

Jackson's DNF learning paper gives a way to use a weak DNF learning algorithm with respect to the uniform distribution to obtain a (strong) DNF learning algorithm. It consists of generalizing a weak parity algorithm to work for any real-valued function (and not only Boolean functions). An important property of the generalized algorithm is that its running time depends polynomially on the L_∞ norm of the function. This algorithm is then used with a boosting algorithm that produces distributions that are *polynomially-close* to the uniform distribution; that is, the distribution function is bounded by $p2^{-n}$ where p is a polynomial in learning parameters (such boosting algorithms are called *p-smooth*). In Jackson's result Freund's boost-by-majority algorithm [Fre90] is used to produce distribution functions bounded by $O(\epsilon^{-(2+\rho)})$ (for arbitrarily small constant ρ). More recently, Klivans and Servedio have observed [KS03] that a later Freund's algorithm [Fre92] produces distribution functions bounded by $\tilde{O}(\epsilon)$, thereby improving the dependence of running time and sample complexity on ϵ. This improvement together with improved weak DNF learning algorithm due to Bshouty *et al.* gives DNF learning algorithm that runs in $\tilde{O}(ns^6/\epsilon^2)$ time and has sample complexity of $\tilde{O}(ns^4/\epsilon^2)$.

Remark 13 *Bshouty* et al. *claimed sample complexity of $\tilde{O}(ns^2/\epsilon^2)$ based on erroneous assumption that sample points for weak DNF learning can be reused across boosting stages. A distribution function $\mathcal{D}_i$ in i-th stage depends on hypotheses produced in previous stages. The hypotheses depend on random sample points and therefore in i-th stage the same set of sample points cannot be considered as chosen randomly and independently of $\mathcal{D}_i$ [Jac04]. This implies that new and independent points have to be sampled for each boosting stage and increases the sample complexity of the algorithm by Bshouty* et al. *by a factor of $O(s^2)$.*

We now briefly describe the generalization of weak parity learning and the boosting step, stressing only the points relevant to our improvements. Let f be the target DNF expression of size s. Lemma 10 states that f has an $\Omega(1/s)$-correlated parity of degree bounded by $O\left(\log\left(sL_\infty(2^n\mathcal{D})\right)\right)$. This implies that function $f(x)2^n\mathcal{D}(x)$ has an $\Omega(1/s)$-heavy Fourier coefficient of degree bounded by $O\left(\log\left(sL_\infty(2^n\mathcal{D})\right)\right)$. Therefore one can expect that weak parity algorithm `WeakDNF-`$\mathcal{U}$ applied to function $f2^n\mathcal{D}$ should find the desired parity. By revisiting the proof of Theorem 11 we can see that the only concern is Equation 2 in which we used the fact that the random variable $f(y) \in \{-1,+1\}$ has variance $\sigma^2 \leq 1$. This is likely to be wrong for random variable $f(y)2^n\mathcal{D}(y)$. Instead we can derive that (expectations are by default for x chosen randomly from $\mathcal{U}$)

$$\begin{aligned} \sigma^2 &= \mathbf{Var}(f(x)2^n\mathcal{D}(x)) = \mathbf{E}[(f(x)2^n\mathcal{D}(x))^2] - \mathbf{E}^2[f(x)2^n\mathcal{D}(x)] \qquad (4) \\ &\leq L_\infty(2^n\mathcal{D}(x))\mathbf{E}[2^n\mathcal{D}(x)] - \mathbf{E}^2[2^n\mathcal{D}(x)] < L_\infty(2^n\mathcal{D}(x)) \qquad (5) \end{aligned}$$

This bound on variance relies essentially on the fact that $\mathcal{D}(x)$ is a distribution[5] and is better than $L_\infty^2(2^n\mathcal{D}(x))$ bound for an unrestricted function $\mathcal{D}(x)$ that was used in analysis of previous weak DNF learning algorithms [Jac94, BJT99]. The only thing that has to be done to offset this higher variance is to use a larger sample space (by a factor of $L_\infty(2^n\mathcal{D}(x))$). This yields the following weak DNF learning result.

5. Actual $\mathcal{D}(x)$ given to a weak learner will be equal to $c\mathcal{D}'(x)$ where $\mathcal{D}'(x)$ is a distribution and c is a constant in $[2/3, 4/3]$ [BJT99]. This modifies the bound above by a small constant factor.

Theorem 14 *There exist an algorithm* `WeakDNF` *that for a Boolean function f of DNF-size s and any distribution $\mathcal{D}(x)$, given n, s, and access to MEM(f), with probability at least $1/2$, finds a $(\frac{1}{2} - \Omega(\frac{1}{s}))$-approximator to f with respect to $\mathcal{D}$. Furthermore,* `WeakDNF`

- *runs in time $\tilde{O}\left(ns^2 L_\infty(2^n \mathcal{D}(x)) + t_{\mathcal{D}}\right)$ where $t_{\mathcal{D}}$ is a bound on the time required to estimate $\mathcal{D}(x)$ on all the points used as MQs of* `WeakDNF`*;*
- *asks $\tilde{O}\left(s^2 \log^2 n \cdot L_\infty(2^n \mathcal{D}(x))\right)$ non-adaptive MQs;*
- *returns a parity function on at most $O(\log(s \cdot L_\infty(2^n \mathcal{D}(x)))$ variables or its negation.*

5.1 Optimized Boosting

The next modification specifically addresses the bound $t_{\mathcal{D}}$. Evaluation of distribution function $\mathcal{D}_i(x)$ at boosting stage i usually involves evaluation of $i-1$ previous hypotheses on x and therefore, in a general case, for a sample of size q will require $\Omega(i \cdot q)$ steps making the last stages of boosting noticeably slower. We show that, in fact, for most known boosting methods the complexity of boosting a weak learner based on Levin's algorithm (in particular the weak learning algorithm by Bshouty *et al.* and `WeakDNF`) can be significantly reduced. The idea is to use the fact that most boosting algorithms compose weak hypotheses linearly, the samples come from a linear subspace of low dimension, and parities are linear functions.

Lemma 15 *Let $\{c_1, c_2, \ldots, c_i\}$ be a set of vectors in $\{0,1\}^n$ of Hamming weight at most w; $\bar{\alpha} \in \mathbb{R}^i$ be a real-valued vector, and R be a m-by-n 0-1 matrix. Then the set of pairs*

$$S = \{\langle p, \sum_{j \le i} \alpha_j \chi_{c_j}(pR) \rangle \mid p \in \{0,1\}^m\}$$

can be computed in time $\tilde{O}(i \cdot w \log n + 2^m)$.

Proof We define $g(x) = \sum_{j \le i} \alpha_j \chi_{c_j}(x)$ and for $p \in \{0,1\}^m$ we define $g_R(p) = g(pR)$ (as in Sect. 4). Our goal is to find the values of function g on all the points of some k-dimensional subspace of $\{0,1\}^n$. The function is given as a linear combination of parities, or in other words, we are given its Fourier transform. Hence the problem is simply to compute the inverse Fourier transform or g. This task can be performed in $O(k2^k)$ steps using the FFT algorithm. Naturally, the transform has to be done from the Fourier coefficients of g_R and not g (as we are given). But the relation between the complete and restricted transforms is simple and follows from the formula below.

$$g_R(p) = \sum_{j \le i} \alpha_j \chi_{c_j}(pR) = \sum_{j \le i} \alpha_j \chi_{c_j R^T}(p) = \sum_{z \in \{0,1\}^m} \left[\Big(\sum_{j \le i;\ c_j R^T = z} \alpha_j \Big) \chi_z(p) \right]$$

Hence $\widehat{g_R}(z) = \sum_{j \le i;\ c_j R^T = z} \alpha_j$. To compute the Fourier transform of g_R we need to compute $c_j R^T$ for each $j \le i$ and sum the ones that correspond to the same z. Given that each c_j is of Hamming weight w, $c_j R^T$ can be computed in $O(wm \log n)$ steps. Therefore the computation of the Fourier transform and the inversion using the FFT algorithm will take $O(i \cdot w \log n + m2^m)$ steps. ■

Corollary 16 *Let $\{b_1\chi_{c_1}, b_2\chi_{c_2}, \ldots, b_i\chi_{c_i}\}$ be a set of hypotheses returned by* `WeakDNF` *in i stages of a certain L-smooth boosting algorithm ($b_j \in \{-1,+1\}$ is a sign of χ_{c_j}); $\bar{\alpha} \in \mathbb{R}^i$ be a real-valued vector; and W be a set of queries for the $(i+1)$-th execution of* `WeakDNF`*. Then the set of pairs*

$$S = \{\langle y, \sum_{j \le i} \alpha_j b_j \chi_{c_j}(z)\rangle \mid z \in W\}$$

can be computed in time $\tilde{O}(i + s^2 L \log^2 n)$.

Proof As can be seen from the proof of Theorem 11, `WeakDNF` asks queries on set $Y = \{pR \mid p \in \{0,1\}^m\}$ for a randomly chosen R and $2^m = \tilde{O}(s^2 L \log^2 n)$ to compute the Fourier transform of $(f 2^n \mathcal{D}_{i+1})_R$ and then for each query y of ae.naMQ parity learning algorithm it computes the Fourier transform of $(f 2^n \mathcal{D}_{i+1})_{y,R}$ by asking queries on points in the set $Y_y = \{z \oplus y \mid z \in Y\}$. The set Y_y is a subset of linear subspace of dimension $m+1$ spanned by the rows of R and vector y. Therefore by using Lemma 15 on subspace Y and then on each Y_y we can compute the set S in $\tilde{O}(i + s^2 L \log^2 n)$ time. ∎

To apply these observations to the computation of distribution function $\mathcal{D}_i$ generated while learning DNF we need to look closer at Freund's boosting algorithm $\mathtt{B}_{\mathtt{Comb}}$ [Fre92, KS03]. It is based on a combination of two other boosting algorithms. The first one `F1` is used to boost from accuracy $\frac{1}{2} - \gamma$ to accuracy $1/4$. The output of the first booster is used as a weak learner by the second boosting algorithm $\mathtt{B}_{\mathtt{Filt}}$. Each of the executions of `F1` has $O(\gamma^{-2})$ stages and $\mathtt{B}_{\mathtt{Filt}}$has $O(\log(1/\epsilon))$ stages. Accordingly, the distribution function can be decomposed into $\mathcal{D}_{i,j}(x) = \mathcal{D}_i^{\mathtt{Filt}} \cdot \mathcal{D}_j^{\mathtt{F1}}$. In both boosting algorithms by Freund the weight of a point equals to $w_i(N(x))/\alpha$ where $N(x)$ is the number of previous hypotheses that are correct on x, w_i is a certain real-valued function, and α is a normalization factor independent of x. Therefore the only information about the previous hypotheses that is needed to compute $\mathcal{D}_j^{\mathtt{F1}}$ is the number of them that are correct on x. Let $b_1\chi_{c_1}, b_2\chi_{c_2}, \ldots, b_{j-1}\chi_{c_{j-1}}$ be the hypotheses generated by previous stages of `F1`. Then $N(x) = \frac{f(x)(\sum_{l \le j-1} b_l \chi_{c_l}(x)) + j - 1}{2}$, that is, given $\sum_{l \le j-1} b_l \chi_{c_l}(x)$ and $f(x)$, $N(x)$ can be computed in $O(1)$ steps. Therefore Cor. 16 implies that $\mathcal{D}_j^{\mathtt{F1}}(x)$ for all the points needed by `WeakDNF` can be computed in $\tilde{O}(s^2 \log^2 n/\epsilon)$ steps (values of f for all the points in the sample are available in `WeakDNF`).

Let $h_1, h_2, \ldots, h_{i-1}$ be the previous hypotheses needed for computation of $\mathcal{D}_i^{\mathtt{Filt}}$. For each $l \le i-1$, h_l is output of `F1` or a random coin flip. Majority of $O(s^2)$ parities (or their negations) is simply the sign of their sum. Hence by Cor. 16, $h_l(x)$ for all the points in the sample for `WeakDNF` can be computed in $\tilde{O}(s^2 \log^2 n/\epsilon)$ time. $\mathtt{B}_{\mathtt{Filt}}$has $O(\log(1/\epsilon))$ stages and therefore all the previous hypotheses can be computed in $\tilde{O}(s^2/\epsilon)$ time and consequently $\mathcal{D}_i^{\mathtt{Filt}}(x)$ can be computed in $\tilde{O}(s^2/\epsilon)$ time.

To finish the analysis of the DNF learning algorithm we describe the computation of the normalization factor α for each of the distributions $\mathcal{D}_i^{\mathtt{Filt}}$ and $\mathcal{D}_j^{\mathtt{F1}}$. For $\mathcal{D}_i^{\mathtt{Filt}}(x)$ this factor equals $\mathbf{E}_{\mathcal{U}}[w_i(N(x))]$ and needs to be estimated within $c_1\epsilon$ for a constant c_1. By Chernoff bounds this can be done (with confidence $1-\delta$) using a random sample of size $O(\frac{1}{\epsilon}\log\frac{1}{\delta})$. All the previous hypotheses can be estimated on this sample in time $\tilde{O}(s^2/\epsilon)$ and therefore the total running time of each boosting stage will not grow (asymptotically).

For $\mathcal{D}_j^{\mathtt{F1}}$ called in i-th iteration of $\mathtt{B}_{\mathtt{Filt}}$the normalization factor is defined to be

$$\alpha = \mathbf{E}_{\mathcal{D}_i^{\mathtt{Filt}}}[w_j(N(x))] = \mathbf{E}_{\mathcal{U}}[2^n \cdot \mathcal{D}_i^{\mathtt{Filt}} \cdot w_j(N(x))]$$

and has to be estimated within a constant c_2. In this case the straightforward estimation via Hoeffding bound will be too expensive since estimation of $\mathcal{D}_i^{\mathtt{Filt}} \cdot w_j(N(x))$ on independent points is more "expensive" than on pairwise independent sample. As we have previously observed, $2^n \cdot \mathcal{D}_i^{\mathtt{Filt}}(x)$ is a distribution (or some constant multiple of a distribution) and is bounded by $\tilde{O}(\frac{1}{2^n \epsilon})$. On the other hand, $w_j(N(x))$ is defined to be in $[0, 1]$ range. Therefore, analogously to (4) we can conclude that the variance of $2^n \cdot \mathcal{D}_i^{\mathtt{Filt}} \cdot w_j(N(x))$ is bounded by $\tilde{O}(\frac{1}{\epsilon})$. Hence we can take a pairwise sample S_1 that gives an estimate of α within c_2 with probability at least 3/4. By Bienaymé-Chebyshev's inequality 1 a sample of size $\tilde{O}(\frac{1}{\epsilon})$ is sufficient. We take l independent samples like this and denote the generated estimates by $\alpha_1, \alpha_2, \ldots, \alpha_l$. When $l = O(\log(1/\delta))$ with probability at least $1 - \delta$ more than a half of the estimates will be within c_2 of true value α. Therefore if we take the median of the set $\{\alpha_1, \alpha_2, \ldots, \alpha_l\}$ it will necessarily approximate α within c_2. From the description and Cor. 16 it is clear that this estimation will take $\tilde{O}(s^2/\epsilon)$ time and hence will not increase the running time of each boosting stage.

Remark 17 *While the analysis of the speedup was done for Freund's* $\mathtt{B}_{\mathtt{Comb}}$ *booster the same idea works for any other booster in which estimation of new weight function is based on a linear combination of previous hypotheses. In particular, for the other known boosting algorithms that produce smooth distributions:* `SmoothBoost` *by Servedio [Ser03] and* `AdaFlat` *by Gavinsky [Gav03].*

Altogether we have obtained a learning algorithm for DNF expressions with the following properties.

Theorem 18 *There exists an algorithm* `AENALearnDNF` *that for any Boolean function f of DNF-size s, given n, s, ϵ, and access to MEM(f), with probability at least 1/2, finds an ϵ-approximator to f with respect to $\mathcal{U}$. Furthermore,* `AENALearnDNF` *runs in time $\tilde{O}\left(ns^4/\epsilon\right)$ and asks $\tilde{O}\left(s^4 \log^2 n/\epsilon\right)$ non-adaptive MQs.*

The improvements to the algorithm by Bshouty *et al.* are summarized below.

- The use of attribute-efficient weak learning improves the total sample complexity from $\tilde{O}\left(ns^4/\epsilon^2\right)$ to $\tilde{O}\left(s^4 \log^2 n/\epsilon^2\right)$ and the same running time is achieved without assumptions on the MQ oracle (see Theorem 12).

- Faster computation of distribution functions used in boosting improves the total running time from $\tilde{O}\left(ns^6/\epsilon^2\right)$ to $\tilde{O}\left(ns^4/\epsilon^2\right)$ (see Corollary 16).

- Tighter estimation of variance improves the dependence of running time and sample complexity on ϵ from $1/\epsilon^2$ to $1/\epsilon$ (equation 4).

5.2 Handling Noise

Now we would like to show that our DNF learning algorithm can be easily converted to a noise tolerant one. Noise model we consider is *random persistent classification noise* introduced by by Goldman, Kearns and Shapire [GKS93]. In this model, the answer to a query at each point x is corrupted with probability η (the noise rate). However, if the membership oracle was already queried about the value of f at some specific point x or x was already generated as a random example, the returned label has the same value as in the first occurrence (i.e., in such a case the noise persists and is not purely random). This persistency is necessary to prevent the learner from removing the noise by repeatedly asking for the label of the same point. However, if the learner does not ask for the label of a point more than once then this noise can be treated as usual random classification noise.

We start by observing that in the algorithm `AENALearnDNF` all the points that are given to the MQ oracle are chosen uniformly and the points that are used in different executions of `WeakDNF` are independent. As can be seen from the proof of Theorem 11, the generated points are of the form $pR \oplus y$ where R is a randomly and uniformly chosen matrix and y is chosen randomly according to $\mathcal{D}_{\frac{1}{4k}}$ or equal to 0^n. If two points y_1 and y_2 are chosen randomly from $\mathcal{D}_{\frac{1}{4k}}$, or one of them equals 0^n then $\mathbf{Pr}[y_1 = y_2] \leq (1 - \frac{1}{4k})^n \leq e^{-n/(4k)}$. If $y_1 \neq y_2$ or $y_1 = y_2 = 0^n$ then two points $p_1 R \oplus y_1$ and $p_2 R \oplus y_2$ are different samples only if $p_1 \neq p_2$. They are equal if $(p_1 \oplus p_2)R = y_1 \oplus y_2$. For a randomly chosen matrix R with m rows, this happens with probability at most 2^{m-n}. Both k and m are logarithmic in learning parameters and the total number of points is polynomial. Therefore the total probability of asking an MQ for the same point is exponentially small. Hence in the following analysis we assume that we are dealing with random and independent classification noise.

5.2.1 Boosting Weak Parity Learning Algorithm in the Presence of Noise

The main part of the modification is to show any weak parity learning algorithm that can handle functions whose values are random variables can be boosted to a DNF learning algorithm in the presence of noise. Our method can be applied in more general setting. In particular, it could be used to prove that Jackson's original algorithm is resistant to persistent noise in MQs and was recently used to produce a noise tolerant DNF learning algorithm by Feldman *et al.* [FGKP06].

The goal of a weak DNF learning algorithm is to find a parity correlated with the function $g(x, f(x)) = 2^n \mathcal{D}(x, f(x)) f(x)$ given the oracle for f with noise of rate $\eta < 1/2$ and a weak parity learning algorithm that can handle functions whose values are random variables.

We use the following observation due to Bshouty and Feldman [BF02]. For any real-valued function $\psi(x, b)$

$$\psi(x, f(x)) = \psi(x, -1)\frac{1 - f(x)}{2} + \psi(x, 1)\frac{1 + f(x)}{2} =$$

$$\frac{1}{2}((\psi(x, 1) - \psi(x, -1))f(x) + \psi(x, 1) + \psi(x, -1)) \ .$$

Let $\Phi^\eta(x)$ denote a random variable that is equal to $f(x)$ with probability $1-\eta$ and to $-f(x)$ with probability η. This is exactly the noisy version of $f(x)$. Then

$$\mathbf{E}_{x,b\sim\Phi^\eta(x)}[\frac{1}{2}(\psi(x,1)-\psi(x,-1))\cdot b] = (1-2\eta)\mathbf{E}_x[\frac{1}{2}(\psi(x,1)-\psi(x,-1))f(x)] \ ,$$

and therefore we can offset the effect of noise in $g(x,f(x))$ as follows.

$$\begin{aligned}[\widehat{g(x,f(x))}](a) &= \mathbf{E}[g(x,f(x))\chi_a(x)] \\ &= \frac{1}{2}(\mathbf{E}_x[(g(x,1)-g(x,-1))\chi_a(x)f(x)] + \mathbf{E}_x[(g(x,1)+g(x,-1))\chi_a(x)]) \\ &= \frac{1}{2}(\frac{1}{1-2\eta}\mathbf{E}_{x,b\sim\Phi^\eta(x)}[(g(x,1)-g(x,-1))\chi_a(x)\cdot b] + \mathbf{E}_x[(g(x,1)+g(x,-1))\chi_a(x)]) \\ &= \mathbf{E}_{x,b\sim\Phi^\eta(x)}\left[\frac{1}{2}\left(\frac{1}{1-2\eta}(g(x,1)-g(x,-1))\cdot b + g(x,1)+g(x,-1)\right)\chi_a(x)\right]\end{aligned}$$

Therefore if we denote $\Psi(x)$ to be the random variable equal to

$$\Psi(x) = \frac{1}{2}\left(\frac{1}{1-2\eta}(g(x,1)-g(x,-1))\cdot\Phi^\eta(x) + g(x,1)+g(x,-1)\right)$$

then for every a, $[\widehat{g(x,f(x))}](a) = \hat{\Psi}(a)$ and therefore we can find a θ-heavy Fourier coefficient of $g(x,f(x))$ by finding a θ-heavy Fourier coefficient of Ψ. For analyzing the performance of a weak parity learner on Ψ it is important to note that $L_\infty(\Psi) \le \frac{2^{n+1}}{1-2\eta}L_\infty(\mathcal{D})$ and $\mathbf{Var}(\Psi) \le \frac{2^{n+1}}{(1-2\eta)^2}L_\infty(\mathcal{D})$.

We now need to prove that our weak parity learning algorithm can handle functions whose values are random variables. Our analysis treats the value $2^n\mathcal{D}(x,f(x))f(x)$ as a random variable produced by choosing x randomly with respect to the uniform distribution. Therefore the fact that given x, $\Psi(x)$ is itself a random variables does not change the algorithm as long as we account for the changed variance of the variable. This is done in the same way as it was done in the proof of Theorem 14. Altogether this gives us the following algorithm for learning DNF expressions with persistent classification noise in MQs.

Theorem 19 *There exists an algorithm* $\texttt{AENALearnDNF}^\eta$ *that for any Boolean function f of DNF-size s, given n, s, ϵ, and access to MEM(f) corrupted by random persistent classification noise of rate η, with probability at least $1/2$, finds an ϵ-approximator to f with respect to $\mathcal{U}$. Furthermore,* $\texttt{AENALearnDNF}^\eta$ *runs in time $\tilde{O}\left(ns^4/(\epsilon(1-2\eta)^2)\right)$ and asks $\tilde{O}\left(s^4\log^2 n/(\epsilon(1-2\eta)^2)\right)$ non-adaptive MQs.*

6. Acknowledgments

We thank Leslie Valiant for his advice and encouragement of this research. We are grateful to Jeffrey Jackson for discussions and clarifications on the DNF learning algorithm of Bshouty *et al.* We also thank Alex Healy, Dmitry Gavinsky and anonymous COLT reviewers for valuable comments and proofreading of the earlier version of this paper.

References

[BF02] N. Bshouty and V. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machince Learning Research*, 2:359–395, 2002.

[BH98] N. Bshouty and L. Hellerstein. Attribute efficient learning with queries. *Journal of Computer and System Sciences*, 56:310–319, 1998.

[BHL95] A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. *JCSS*, 50:32–40, 1995.

[BJT99] N. Bshouty, J. Jackson, and C. Tamon. More efficient PAC learning of DNF with membership queries under the uniform distribution. In *Proceedings of COLT '99*, pages 286–295, 1999.

[BMOS03] N. Bshouty, E. Mossel, R. O'Donnell, and R. Servedio. Learning DNF from random walks. In *Proceedings of FOCS '03*, pages 189–199, 2003.

[CT65] J. Cooley and J. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Computat.*, 19:297–301, 1965.

[Dam98] P. Damaschke. Adaptive versus nonadaptive attribute-efficient learning. In *Proceedings of STOC '98*, pages 590–596, 1998.

[FGKP06] V. Feldman, P. Gopalan, S. Khot, and A. K. Ponnuswami. New results for learning noisy parities and halfspaces. *Electronic Colloquium on Computational Complexity (ECCC)*, (059), 2006.

[FKKM97] M. Farach, S. Kannan, E. Knill, and S. Muthukrishnan. Group testing problems in experimental molecular biology. In *Proceedings of Sequences '97*, 1997.

[Fre90] Y. Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 202–216, 1990.

[Fre92] Y. Freund. An improved boosting algorithm and its implications on learning complexity. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 391–398, 1992.

[Gav03] D. Gavinsky. Optimally-smooth adaptive boosting and application to agnostic learning. *J. Mach. Learn. Res.*, 4:101–117, 2003.

[GKS93] S. Goldman, M. Kearns, and R. Schapire. Exact identification of read-once formulas using fixed points of amplification functions. *SIAM J. Comput.*, 22(4):705–726, 1993.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of STOC '89*, pages 25–32, 1989.

[GLR99] D. Guijarro, V. Lavin, and V. Raghavan. Exact learning when irrelevant variables abound. In *Proceedings of EuroCOLT '99*, pages 91–100, 1999.

[GTT99] D. Guijarro, J. Tarui, and T. Tsukiji. Finding relevant variables in PAC model with membership queries. *Lecture Notes in Artificial Intelligence*, 1720:313 – 322, 1999.

[Hof99] T. Hofmeister. An application of codes to attribute-efficient learning. In *Proceedings of EuroCOLT*, pages 101–110, 1999.

[Jac94] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. In *Proceedings of STOC '94*, pages 42–53, 1994.

[Jac04] J. Jackson. Personal communication, 2004.

[JSS97] J. Jackson, E. Shamir, and C. Shwartzman. Learning with queries corrupted by classification noise. In *Proceedings of the Fifth Israel Symposium on the Theory of Computing Systems*, page 45. IEEE Computer Society, 1997.

[KS03] A. Klivans and R. Servedio. Boosting and hard-core set construction. *Machine Learning*, 51(3):217–238, 2003.

[KS04] A. Klivans and R. Servedio. Toward attribute efficient learning of decision lists and parities. In *Proceedings of COLT '04*, pages 234–248, 2004.

[Lev93] L. Levin. Randomness and non-determinism. *Journal of Symbolic Logic*, 58(3):1102–1103, 1993.

[Man94] Y. Mansour. *Learning Boolean functions via the Fourier transform*, pages 391–424. 1994.

[Ser03] R. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

[UTW97] R. Uehara, K. Tsuchida, and I. Wegener. Optimal attribute-efficient learning of disjunction, parity, and threshold functions. In *Proceedings of EuroCOLT '97*, pages 171–184, 1997.

[Val84] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Val94] L. Valiant. *Circuits of the Mind*. Oxford University Press, 1994.

[Val00] L. Valiant. A neuroidal architecture for cognitive computation. *Journal of ACM*, 47(5):854–882, 2000.

[Val05] L. Valiant. Knowledge infusion (unpublished manuscript). 2005.