

Learning Threshold Functions with Small Weights using Membership Queries

Elias Abboud
Research Center
Ibillin Elias College
P.O. Box 102
Ibillin 30012

Nader Agha
Research Center
Ibillin Elias College
P.O. Box 102
Ibillin 30012

Nader H. Bshouty
Department of Computer Science
Technion
Haifa 32000
email: bshouty@cs.technion.ac.il

Nizar Radwan
Department of Mathematics
Technion
Haifa 32000
email: radwan@math.technion.ac.il

Fathi Saleh
Research Center
Ibillin Elias College
P.O. Box 102
Ibillin 30012 *

Abstract

We study the learnability of Threshold functions with bounded weights using membership queries only. We show that the class C_t of Threshold functions with positive integer weights that are less or equal

*This research was done in a discussion seminar that was opened in the spring 1997 at the research center of Ibillin Elias College.

to t is learnable with $n^{O(t^5)}$ membership queries. We also provide a lower bound of $\Omega(n^t)$ for the number of membership queries required to learn this class. We also show that learning the class with weights from $\{-1, 0, 1\}$ requires at least $\Omega(2^n)$ membership queries.

1 Introduction

One of the interesting models in learning is the exact learning model with queries [A88, L88]. In this model the *learning algorithm* can ask queries about an unknown *target function* $f : \{0, 1\}^n \rightarrow \{0, 1\}$ that belongs to a class of functions C . The learning algorithm is required to identify the target function f (or an equivalent one from C) in time polynomial in the number of variables. If such an algorithm exists then we say that the class C is *learnable*. The queries considered in the literature are the *equivalence queries* and the *membership queries*. An equivalence query takes as input a hypothesis $h \in C$ and returns YES if h is equivalent to f and NO with a *counterexample* y otherwise. The counterexample y satisfies $f(y) \neq h(y)$. A membership query takes as input an assignment $z \in \{0, 1\}^n$ and returns the value of $f(z)$.

A *Threshold function* f with integer weights a_1, \dots, a_n and integer threshold b is a function from $\{0, 1\}^n$ to $\{0, 1\}$ such that

$$f = \begin{cases} 1 & a_1x_1 + a_2x_2 + \dots + a_nx_n \geq b \\ 0 & \text{otherwise.} \end{cases}$$

Maass and Turan [MT89] shows that the class of Threshold functions is learnable from equivalence queries only. It is very easy to prove that this class is not learnable from membership queries only.

In this paper we place some restrictions on the weights and show that with these restrictions, learnability with membership queries is possible. We show that the class of threshold functions C_t with weights that are positive integers that are less than or equal to a constant t is learnable. Our learning algorithm returns a Threshold function from C_t . We also show that this class is not learnable for nonconstant t . For thresholds with negative weights we show that the class of thresholds with weights in $\{0, 1, -1\}$ is not learnable from membership queries only.

2 Preliminaries

In this section we give some preliminary results that will be used to explain our algorithm.

2.1 Classes and Learnability

Let C be a class of formulas that represents boolean functions over $\{x_1, \dots, x_n\}$. An *assignment* x is $x \in \{0, 1\}^n$. As in complexity theory we will assume here that we have a sequence of classes C^n . Then, when we talk about complexity being polynomial in n we mean that there are constant c and d such that for every $n \geq 2$ the algorithm runs at most $n^c + d$ steps for learning C^n . We say that C is *closed under projection* if for any $f \in C$ and any variable x_i and value $\xi \in \{0, 1\}$, the projection $f|_{x_i \leftarrow \xi}$ of f on $x_i = \xi$ is a function in C . We say that C is *solvable* if there is an algorithm that on inputs $f, g \in C$ runs in polynomial time in n and outputs YES if f is equivalent to g and NO with an assignment y such that $f(y) \neq g(y)$, otherwise. We say that C is *weakly solvable* if there is an algorithm that on inputs $f, g \in C$ runs in polynomial time in n and outputs YES if f is equivalent to g and NO otherwise.

We first prove the following

Lemma 1 *Let C be a class of boolean formulas. If C is closed under projection and is weakly solvable then it is solvable.*

Proof: This is a standard searching reduction. We run the algorithm that decides whether f is equivalent to g . If the answer is YES then we are done. Otherwise, we run the algorithm for the projections $f|_{x_1 \leftarrow 0}$ and $g|_{x_1 \leftarrow 0}$. The new input is legal because the class is projection closed. If the answer is YES then we know that $f|_{x_1 \leftarrow 1}$ and $g|_{x_1 \leftarrow 1}$ are not equivalent. In both cases we recursively run the algorithm until we get an assignment y such that $f(y) \neq g(y)$. \square

We say that C is *learnable in time l* with membership queries (and equivalence queries) if there is an algorithm that on input n and with a membership (and equivalence) oracle to $f \in C$ runs in time $l(n)$ and output h that is equivalent to f . We say that C is *learnable* with membership queries (and equivalence queries) if it is learnable in polynomial time. We say that the learning algorithm is *proper* if the output hypothesis h is from C .

The following lemmas will be useful for the lower bounds and for our algorithm

Lemma 2 *Let C be a class of formulas. If C is learnable in time l with membership queries then it is solvable in time $O(lv)$ where v is the time needed to evaluate a function $f \in C$ in some point x .*

Proof: Suppose C is learnable in time l . Given two functions f and g in C . We run the algorithm that learns C and answer queries by substituting in f . Then we take all the assignments that are used in the membership queries for learning f and substitute them in g . It is clear that $f \equiv g$ if and only if $f = g$ on those assignments. \square

Lemma 3 *Let C be a class of formulas. Let $H \supset C = C^{(1)} \cup \dots \cup C^{(m)}$. If H is solvable in time l and $C^{(i)}$ is learnable with membership queries with a learning algorithm that output a hypothesis from H in time $l^{(i)}$ then C is learnable with membership queries with a learning algorithm that output a hypothesis from H in time*

$$m(l + \max_i l^{(i)}).$$

Proof: We run the algorithms that learn $C^{(i)}$ in parallel. One step in each algorithm and at most $\max_i l^{(i)}$ steps for each algorithm. Each time we have two outputs $h_1, h_2 \in H$ we run an algorithm that test if they are equivalent. If they are equivalent then we keep one of them and if they are not then we have an assignment z such that $h_1(z) \neq h_2(z)$. We ask the value of the target f on z and keep the h_i that is consistent. Because $C = C^{(1)} \cup \dots \cup C^{(m)}$, eventually we will have only one function. This function must be equivalent to the target. \square

2.2 Disjoint Sum Functions

In this section we give a generalization of the class of Threshold functions with small weights (introduced in the next section). Some of the results of this paper will also be true for this class.

The t -disjoint sum boolean class \mathcal{DS}_t is the class of all boolean formulas of the form $\varphi(X_0, \dots, X_t)$ where $\varphi : \{0, \dots, n\}^t \rightarrow \{0, 1\}$, $X_i = \sum_{j \in S_i} x_j$ (arithmetic sum) and $\{S_i\}$ is a partition of $\{1, \dots, n\}$ (disjoint sets). We will assume that φ is independent of X_0 . So X_0 will be the sum of all the irrelevant variables. Notice that the class of 1-disjoint sum is the class of symmetric boolean functions. When $\{S_i\}$ is not a partition (not necessarily disjoint) then we call the function t -sum function. The class of all t -sum boolean functions is denoted by \mathcal{S}_t .

It is also easy to see that $\mathcal{DS}_t \subset \mathcal{S}_t$ and

$$\mathcal{S}_t \subseteq \mathcal{DS}_{\min(2^t, n)}.$$

We now show that

Lemma 4 *The class of t -sum boolean functions is solvable in time $n^{O(t)}$.*

Proof: In Theorem 11 we will show that any t -sum function can be changed to an equivalent automaton of size n^{t+1} . Since the class of automata is solvable the lemma follows. \square

The following lemma will be used in this paper

Lemma 5 *Let $f = \varphi(X_0, X_1, \dots, X_t)$ be a t -disjoint sum function but not a $t - 1$ -disjoint sum function. If $f = \varphi'(X'_0, \dots, X'_{t'})$ then $X'_0, \dots, X'_{t'}$ are subsums of X_0, X_1, \dots, X_t . That is X_i are sums of X'_j s.*

Proof: Suppose the claim is not true. Then by permuting the variables x_i and X_j we may assume that $X_1 = x_1 + X'_1$, $X_2 = x_2 + X'_2$ and $X'_1 = x_1 + x_2 + X''_1$. Now notice that $f(0, 1, x_3, \dots, x_n) = f(1, 0, x_3, \dots, x_n)$ and therefore we have $\varphi(n_0, n_1 + 1, n_2, n_3, \dots, n_t) = \varphi(n_0, n_1, n_2 + 1, n_3, \dots, n_t)$ for every n_1 and n_2 . Consequently, for every k $\varphi(n_0, n_1, n_2, \dots, n_t)$ has the same value for all $n_1 + n_2 = k$. Therefore we can write it as $\hat{\varphi}(X_0, X_1 + X_2, X_3, \dots, X_t)$ which is a $t - 1$ -disjoint sum function. We get a contradiction. \square

Using same proof as in Lemma 5 we have

Lemma 6 *Let $f = \varphi(X_0, X_1, \dots, X_t)$ be a t -disjoint sum function that is not a $t - 1$ -disjoint sum function. If $f|_{x_i \leftarrow 0, x_j \leftarrow 1} \equiv f|_{x_i \leftarrow 1, x_j \leftarrow 0}$ then x_i and x_j are both in one X_k for some k .*

Now we prove a better complexity than in Lemma 4 for the class of disjoint k -sum functions

Lemma 7 *The class of t -disjoint sum is solvable in time $O((n + 1)^{t+1t})$.*

Proof: Let $f = \varphi(X_0, X_1, \dots, X_{t_1})$ and $g = \varphi'(X'_0, X'_1, \dots, X'_{t_2})$. If $X'_0, X'_1, \dots, X'_{t_2}$ are subsums of X_0, X_1, \dots, X_{t_1} (or vice versa), then we can find the values of φ' for all possible values of $X'_1 = n_1, \dots, X'_{t_2} = n_{t_2}$ and compare it with φ . This will take at most $n^{t_2} \leq (n + 1)^t$ queries.

Otherwise, there is (say) $X_i = x_1 + x_2 + X'_i$ and $X'_{j_1} = x_1 + X''_{j_1}$ and $X'_{j_2} = x_2 + X''_{j_2}$. By the proof of the Lemma 5 either there is $a \in \{0, 1\}^{n-2}$ such that $f(1, 0, a) \neq f(0, 1, a)$ or we can put X'_{j_1} and X'_{j_2} in one variable. If we find that $f(1, 0, a) = f(0, 1, a)$ for every a (using the above approach for $f|_{x_1 \leftarrow 0, x_2 \leftarrow 1}$ and $f|_{x_1 \leftarrow 1, x_2 \leftarrow 0}$) we combine X'_{j_1} and X'_{j_2} and repeat the above. We have at most $2t$ of those combinations. Otherwise, we find an a such that $f(1, 0, a) \neq f(0, 1, a)$. Since $g(1, 0, a) = g(0, 1, a)$ we must have that either $f(1, 0, a) \neq g(1, 0, a)$ or $f(0, 1, a) \neq g(0, 1, a)$. \square

2.3 Threshold with small Weights

Let C_t be the class of all boolean functions over n variables of the form

$$f = \begin{cases} 1 & a_1x_1 + a_2x_2 + \dots + a_nx_n \geq b \\ 0 & \text{otherwise.} \end{cases}$$

where $a_1, \dots, a_n \in \{0, 1, \dots, t\}$. We will write such a function as $f = [a_1x_1 + \dots + a_nx_n \geq b]$.

It is clear that $C_t \subset \mathcal{DS}_t$ and therefore C_t is solvable in time $O(n^{O(t)})$.

3 Lower Bound

In this section we prove that when t is not a constant then C_t is not learnable.

Theorem 8 *To learn the class C_t we need at least*

$$\Omega\left(\binom{n}{\min(t, n/2)}\right)$$

membership queries.

Proof: Consider the class D_t of all thresholds $[a_1x_1 + \dots + a_nx_n \geq t^2 - t + 1]$ where $a_i \in \{t - 1, t\}$ and exactly t of the coefficients are equal to $t - 1$. The boolean functions in D_t satisfies the following:

1. $f(v) = 0$ for all assignments v that have weight (number of ones in the assignment) that is less than or equal to $t - 1$.
2. $f(v) = 1$ for all assignments v that have weight that is greater than or equal to $t + 1$.
3. $f(v) = 1$ for all assignments v that have weight that is equal to t except for the assignment that is one for all x_i in which $a_i = t - 1$.
4. $f(v) = 0$ for the assignment that is one for all x_i in which $a_i = t - 1$ and 0 elsewhere.

Therefore, learning D_t from membership queries is equivalent to finding this assignment from an oracle that given an assignment of weight t answer YES if it is the correct assignment and NO otherwise. This will take at least $\Omega(|D_t|)$ membership queries (even with a randomized algorithm). \square

We now show that Threshold functions with coefficients from $\{0, 1, -1\}$ are not learnable.

Lemma 9 *To learn Threshold functions with coefficients from $\{0, 1, -1\}$ we need at least $O(2^n)$ membership queries.*

Proof: Notice that the term $x_{i_1} \wedge \dots \wedge x_{i_j} \bar{x}_{i_{j+1}} \wedge \dots \wedge \bar{x}_{i_k}$ is the Threshold function $[x_{i_1} + \dots + x_{i_j} - x_{i_{j+1}} - \dots - x_{i_k} \geq j]$. Now the bound follow from the lower bound for learning terms. \square

Since the class in Theorem 8 is a subset of the class of 2-disjoint sum we have the following.

Corollary 10 *The class of monotone 2-disjoint sum is not learnable from membership queries.*

Proof: The class in Theorem 8 is monotone because all coefficients are positive. It is a 2-disjoint sum because it can be written as $\varphi(X_1, X_2)$ where X_1 contains the variables with coefficients equal to t and X_2 is the rest of the variables. Choose $t = n/2$ and the membership query complexity will be $2^{O(n)}$. \square

4 Learning Sum Functions

In the previous section we showed that the class of 2-disjoint sum is not learnable with membership queries only. Here we will show that this class is learnable when the learner can also ask equivalence queries. We will prove the following

Theorem 11 *The class of t -sum boolean functions is learnable from $n^{O(t)}$ membership and equivalence queries.*

Proof: We will show that we can turn any t -sum function into an automaton of size n^t . We then use Angluin's algorithm [A87] for learning automata.

Let $f = \varphi(X_1, \dots, X_t)$. Consider the states at level l ,

$$Q_l = \{(X_1, \dots, X_t)_{|(x_1, \dots, x_{l-1}) \leftarrow v} \mid v \in \{0, 1\}^{l-1}\}.$$

So every state at level l is of the form $(Y_1 + \lambda_1, \dots, Y_t + \lambda_t)$ where Y_i is X_i without the variables x_1, \dots, x_{l-1} and $\lambda_i \leq l - 1$. Therefore, at level l we have at most $l^t < n^t$ states.

The initial state is (X_1, \dots, X_n) . The state $(Y_1 + \lambda_1, \dots, Y_t + \lambda_t)$ at level l will be connected to two states at level l . It is connected to the state $(Y_1 + \lambda_1, \dots, Y_t + \lambda_t)_{|x_l \leftarrow 0}$ with an edge labeled with 0 and to the state $(Y_1 + \lambda_1, \dots, Y_t + \lambda_t)_{|x_l \leftarrow 1}$ with an edge labeled with 1. At level n the state $(\lambda_1, \dots, \lambda_n)$ will be a final state if and only if $\varphi(\lambda_1, \dots, \lambda_n) = 1$. Also every state in level n is connected to a nonfinal state with a loop (a sink state).

Obviously, the string $s \in \{0, 1\}^*$ is accepted by the above automaton if and only if its length is n and $f(s_1, \dots, s_n) = 1$. The size of the automata is at most n^{t+1} . \square

5 The Algorithm

In this section we give an algorithm that learns the class C_t in time $n^{O(t^5)}$.

5.1 The Algorithm for Small b

We will first show that if $f = [a_1x_1 + \dots + a_nx_n \geq b]$ and $b \leq \beta$ or $b \geq \sum_i a_i - \beta$ where β is a small constant then there is an algorithm that runs in time $O(n^{\beta+2})$ uses membership queries and learns a function in \mathcal{DS}_t that is equivalent to f . We then show how to change this function to a function in C_t in constant (depends on β and t) time.

First it is easy to see that if $b \geq \sum_i a_i - \beta$ then the function $\overline{f(\bar{x}_1, \dots, \bar{x}_n)}$ is in C_t with $b \leq \beta$. So it is enough to solve the former case.

Because $b \leq \beta$ all assignments of weight (number of 1s in the assignment) greater than β gives the answer 1. So it is enough to find the value of the function for all assignments of weight at most β . There are at most n^β such assignments. Now using a table T that contains the values of those assignment, the hypothesis

$$h(x) = \begin{cases} T(x) & wt(x) < \beta \\ 1 & \text{otherwise,} \end{cases}$$

is equivalent to the target function. To change h to a function in \mathcal{DS}_t we use Lemma 6. For every two variables x_i and x_j we check if $f|_{x_i \leftarrow 0, x_j \leftarrow 1} \equiv f|_{x_i \leftarrow 1, x_j \leftarrow 0}$. This can be done in time $n^{\beta+2}$. After we have X_0, \dots, X_t we find φ in time $n^t \leq n^\beta$.

We now show how to change this hypothesis to a threshold function in C_t . We will first prove the following.

Lemma 12 *Let $f \equiv \varphi(X_0, X_1, \dots, X_t)$ and $f \equiv [a_1x_1 + \dots + a_nx_n \geq b]$. Suppose (after permuting the variables) $X_1 = x_1 + x_2 + \dots + x_r$. If $r > 2b$ then $f \equiv [a_jx_1 + \dots + a_jx_r + a_{r+1}x_{r+1} + \dots + a_nx_n \geq b]$ for some $j \leq r$. That is, if X_1 contains more than $2b$ variables then we can change the threshold function so that all the variables in X_1 have the same coefficients in f .*

Proof: First notice that because all the variables in X_1 are relevant in the function we have $a_i \neq 0$ for $i = 1, \dots, r$. We will also assume that $a_1 \leq \dots \leq a_r$. Otherwise, we can permute the variables to have this property. Let $j = \lfloor r/2 \rfloor$. Now define $g = [a_jx_1 + \dots + a_jx_r + a_{r+1}x_{r+1} + \dots + a_nx_n \geq b]$. Suppose $f \not\equiv g$. Then there is an assignment c such that $f(c) \neq g(c)$. Suppose $f(c) = 1$ and $g(c) = 0$ (the proof for the other case is similar). We will now concentrate on c_1, \dots, c_r .

Because $g(c) = 0$ we have $a_jc_1 + \dots + a_jc_r < b$ and therefore $q = c_1 + \dots + c_r < b$. Now define $c' = (1^q, 0^{r-q}, c_{r+1}, \dots, c_n)$. Notice that $g(c') = g(c) = 0$ and because $X_1 = x_1 + \dots + x_r$ we also have $f(c') = f(c) = 1$. Now because $g(c') = 0$, $c_j \geq c_i$ for $i \leq \lfloor r/2 \rfloor$ and $\lfloor r/2 \rfloor \geq b > q$ we have

$$\begin{aligned} a_1c'_1 + \dots + a_nc'_n &\leq a_jc'_1 + \dots + a_jc'_r + a_{r+1}c'_{r+1} + \dots + a_nc'_n \\ &\leq b \end{aligned}$$

and therefore, $f(c') = 0$ which gives a contradiction. \square

Now for every X_i with more than $2b$ variables we know that they have the same coefficient and for those with less than $2b$ variables, they may or may not have the same coefficients. We now take the X_i that have more than $2b$ variables. There are at most t of them. Then we take the other variables. There are at most $2bt$ variables and then try all possible coefficients for them. There are t^{2bt+t} possible threshold functions. Notice that since t and b are fixed this number is constant. We take all these threshold functions and using a table find an equivalent one. This solves learning for the case when $b \leq \beta$ and β is constant.

5.2 The Algorithm for any b

In this subsection we give the algorithm for C_t . We will first give the following definition.

Definition 13 For a constant w , two sets $S, R \subseteq \{0, \dots, t\}$ where $S \cup R = \{0, \dots, t\}$, a set of indices $\mathcal{I}_S = \{I_s \mid s \in T\}$ where $I_s \subseteq \{1, \dots, n\}$ with $|I_s| = 2w$ and a set of indices $\mathcal{J}_R = \{J_r \mid r \in R\}$ where $J_r \subseteq \{1, \dots, n\}$ with $|J_r| < 2w$, we define the class $C_t^w[S, \mathcal{I}_S, R, \mathcal{J}_R]$ to be the class of all thresholds $f = [a_1x_1 + \dots + a_nx_n \geq b]$ that satisfy the following:

1. $a_i = s$ for all $i \in I_s$ and $s \in S$.
2. $a_j = r$ for all $j \in J_r$ and $r \in R$.
3. $a_j \neq r$ for all $j \notin J_r$ and $r \in R$.

From the above definition, S contains all the values from $\{0, 1, \dots, t\}$ that appear as coefficients of f at least $2w$ times. The set R is the set of all values from $\{0, 1, \dots, t\}$ that appear as coefficients at most $2w$ times in f . The set I_s contains $2w$ indices i such that $a_i = s$. The set J_r contains all the indices i such that $a_i = r$. The class $C_t^w[S, \mathcal{I}_S, R, \mathcal{J}_R]$ contains all the threshold functions with the above properties.

It is clear that for any w , any threshold function in C_t is in some $C_t^w[S, \mathcal{I}_S, R, \mathcal{J}_R]$ for some S, \mathcal{I}_S, R , and \mathcal{J}_R . Therefore

$$C_t = \bigcup_{S, \mathcal{I}_S, R, \mathcal{J}_R} C_t^w[S, \mathcal{I}_S, R, \mathcal{J}_R].$$

Lemma 14 The number of distinct $S, \mathcal{I}_S, R, \mathcal{J}_R$ is at most

$$n^{O(wt)}.$$

Proof: We have 2^t possible (S, R) with $S \cup R = \{0, \dots, t\}$. Then we have at most $\binom{n}{2w}^t \leq n^{O(wt)}$ ways for choosing \mathcal{I}_S and \mathcal{J}_R . \square

Since C_t is solvable, using Lemma 3 we have the following.

Corollary 15 If $C_t^w[S, \mathcal{I}_S, R, \mathcal{J}_R]$ is learnable in time l then C_t is learnable in time

$$n^{O(wt)}(n^t + l).$$

We now show how to learn $f \in C_t^w[S, \mathcal{I}_S, R, \mathcal{J}_R]$. Let $S = \{s_1, \dots, s_\sigma\}$. We will assume that $I_{s_j} = \{2w(j-1) + 1, \dots, 2wj\}$ (a permutation of the variables in the function). Let $K = \{2w\sigma + 1, \dots, n\}$ be the rest of the indices. For a set of indices L we will write 1_L for the assignment that is one in indices $i \in L$ and zero elsewhere.

Let $I_{s_j}^{\frac{1}{2}} = \{2w(j-1) + 1, \dots, 2w(j-1) + w\}$. This set contains half of the elements in the set I_{s_j} . The first step in the algorithm is to ask membership queries with the following two assignments,

$$x^{(1)} = 1_{\cup_j I_{s_j}^{\frac{1}{2}}}, \quad x^{(2)} = 1_{\cup_j I_{s_j}^{\frac{1}{2}} \cup K}.$$

We will have three cases.

1. $f(x^{(1)}) = 1$ and then $f(x^{(2)}) = 1$.
2. $f(x^{(2)}) = 0$ and then $f(x^{(1)}) = 0$.
3. $f(x^{(1)}) = 0$ and $f(x^{(2)}) = 1$.

In the first case since $f(x^{(1)}) = 1$ we have $s_1w + s_2w + \dots + s_\sigma w \geq b$ and therefore we have $b < wt^2$. In the second case we have $b > \sum a_i - wt^2$. In both cases we can use the algorithm in the previous subsection. This will take n^{wt^2+2} time complexity.

We now show the algorithm for the third case.

Let $W_q = \cup_j I_{s_j}^{\frac{1}{2}} \cup \{2w\sigma + 1, \dots, 2w\sigma + q\}$. From the condition above we have $f(1_{W_0}) = 0$ and $f(1_{W_{n-2w\sigma}}) = 1$. Using membership queries we find the minimal q such that $f(1_{W_q}) = 0$ and $f(1_{W_{q+1}}) = 1$. This is the point where the sum of the coefficients becomes approximately equal to b . We will denote by $S(b)$ the sum of the coefficients on the assignment b . So we have $S(1_{W_q}) < b$ and $S(1_{W_{q+1}}) \geq b$. Also since the coefficients are at most t we have $S(1_{W_{q+1}}) - S(1_{W_q}) \leq t$ and therefore

$$b - t \leq S(1_{W_q}) < b.$$

We now will use the assignment 1_{W_q} and flip some of the bits so the sum of the coefficients becomes as close as possible to b . We use the following result from number theory [N].

Proposition 16 *Let $1 \leq s_1 < s_2 < \dots < s_\sigma \leq t$ be integers where $\gcd(s_1, \dots, s_\sigma) = d$. There exist integers w_1, \dots, w_σ such that $|w_i| \leq d + (\sigma - 1)t \leq t^2$ and*

$$w_1 s_1 + \dots + w_\sigma s_\sigma = d.$$

We now use this Proposition to define a new assignment c that gives a sum of the coefficients equal to $S(c) = S(1_{W_q}) + d$. Notice that $w_1 s_1 + \dots + w_\sigma s_\sigma = d$ so for every i if $w_i > 0$ we flip w_i zeros of $I_{s_i}^{1/2}$ in 1_{W_q} to ones and if $w_i < 0$ then we flip w_i ones of $I_{s_i} \setminus I_{s_i}^{1/2}$ in 1_{W_q} to zeros. Obviously, from the proposition, the new assignments c satisfies $S(c) = S(1_{W_q}) + d$. We now find the value of the function in c . We keep doing the above until the final flip will change the value of the function from 0 to 1. So we will have two assignments c and c' such that,

1. $S(c) - S(c') = d$.
2. $f(c) = 1$ and $f(c') = 0$.

Notice that since $S(1_{W_q}) \geq b - t$ we will do at most $t/d \leq t$ rounds of flipping. Each flipping change the value of at most $|w_i| < t^2$ bits in $I_{s_i}^{1/2}$ and $I_{s_i} \setminus I_{s_i}^{1/2}$ and therefore $w = |I_{s_i}^{1/2}|$ must be at least t^3 .

Now to find the coefficients a_i where $i \in K$, we have three cases

Case I: $i \in J_r$ for some r . Then we know that $a_i = r$.

Case II: $i \notin J_r$ for all r and $c'_i = 1$. Then we know that $a_i \in \{s_1, \dots, s_\sigma\}$. Take c' and flip c'_i to zero and some $c_j = 0$ with $a_j = s_l$ to one. The coefficient a_i is equal to the maximal s_l that didn't change the value of the function. The reason is that $s_q - s_{q-1} > d$ and therefore the last one that didn't change the value of the function is the right coefficient.

Case III: $i \notin J_r$ for all r and $c'_i = 0$. Then we know that $a_i \in \{s_1, \dots, s_\sigma\}$. Take c' and flip c'_i to one and some $c_j = 1$ with $a_j = s_l$ to zero. The coefficient a_i is equal to the minimal s_l that didn't change the value of the function.

5.3 Complexity

We need to choose w large enough so we can do the flipping. First notice that $|w_i| \leq d + (\sigma - 1)t \leq t^2$. In the flipping we do at most t flippings so $w = t^3$ is sufficient. The complexity of the algorithm for small b is now $n^{wt^2+2} = n^{t^5}$. The complexity of the algorithm for the third case is $O(n)$. Therefore, using Proposition 16, the complexity is $n^{O(t^5)}$.

References

- [A88] D. Angluin. Queries and Concept Learning. *Machine Learning*, pp 319–342, 2, 4, 1988.

- [A87] D. Angluin. Learning Regular Sets from Queries and Counterexamples. In *Information and Computation*, 75:87–106, 1987.
- [L88] N. Littlestone, “Learning when Irrelevant Attributes Abound: A New Linear-Threshold Algorithm” *Machine Learning*, Vol. 2, pp. 285–318, 1988.
- [MT89] W. Maass and G. Turan. “On the Complexity of Learning from Counterexamples”, *Proc. of 30th Annu. Symp. on Foundations of Computer Science*, pages 262–273, 1989.
- [N77] W. Narkiewicz. *Number Theory*, Word Science, 1977.