April 1989

Polynomial Learnability of Semilinear Sets

Naoki Abe
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-25.

Polynomial Learnability of Semilinear Sets

Abstract

We characterize learnability and non-learnability of subsets of N^m called 'semilinear sets', with respect to the distribution-free learning model of Valiant. In formal language terms, semilinear sets are exactly the class of 'letter-counts' (or Parikh-images) of regular sets. We show that the class of semilinear sets of dimensions 1 and 2 is learnable, when the integers are encoded in unary. We complement this result with negative results of several different sorts, relying on hardness assumptions of varying degrees - from $P \neq NP$ and $RP \neq NP$ to the hardness of learning DNF. We show that the minimal consistent concept problem is NP-complete for this class, verifying the non-triviality of our learnability result. We also show that with respect to the binary encoding of integers, the corresponding 'prediction' problem is already as hard as that of DNF, for a class of subsets of N^m much simpler than semilinear sets. The present work represents an interesting class of countably infinite concepts for which the questions of learnability have been nearly completely characterized. In doing so, we demonstrate how various proof techniques developed by Pitt and Valiant [14], Blumer et al. [3], and Pitt and Warmuth [16] can be fruitfully applied in the context of formal languages.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-25.

POLYNOMIAL LEARNABILITY OF SEMILINEAR SETS

Naoki Abe

MS-CIS-89-25
LINC LAB 149

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

April 1989

Acknowledgements: This research was supported in part by an IBM graduate fellowship, Office of Naval Research grant NOOO14-87-K-0401, DARPA grant NOOO14-85-K-0018, NSF grants MCS-8219196-CER, IRI84-10413-AO2 and U.S. Army grants DAA29-84-K-0061, DAA29-84-9-0027.

Polynomial Learnability of Semilinear Sets*

Naoki Abe

Department of Computer and Information Science,
University of Pennsylvania, Philadelphia, PA19104.

April 5, 1989

Abstract

We characterize learnability and non-learnability of subsets of N^m called 'semilinear sets', with respect to the distribution-free learning model of Valiant. In formal language terms, semilinear sets are exactly the class of 'letter-counts' (or Parikh-images) of regular sets. We show that the class of semilinear sets of dimensions 1 and 2 is learnable, when the integers are encoded in unary. We complement this result with negative results of several different sorts, relying on hardness assumptions of varying degrees – from $P \neq NP$ and $RP \neq NP$ to the hardness of learning DNF. We show that the minimal consistent concept problem is NP-complete for this class, verifying the non-triviality of our learnability result. We also show that with respect to the binary encoding of integers, the corresponding 'prediction' problem is already as hard as that of DNF, for a class of subsets of N^m much simpler than semilinear sets. The present work represents an interesting class of countably infinite concepts for which the questions of learnability have been nearly completely characterized. In doing so, we demonstrate how various proof techniques developed by Pitt and Valiant [14], Blumer et al. [3], and Pitt and Warmuth [16] can be fruitfully applied in the context of formal languages.

1 Introduction

We consider the problem of learning semilinear subsets of N^m, with respect to the distribution-free learnability model of Valiant [18]. Semilinear sets are finite unions of linear subsets of N^m, where a linear set is the set of values assumed by a linear integral formula of the form:

$$v_0 + c_1 \cdot v_1 + ... + c_k \cdot v_k$$

*The work reported herein is supported in part by an IBM graduate fellowship awarded to the author and by the Office of Naval Research under contract number N00014-87-K-0401.

as the c_i's vary over natural numbers, where the v_i's are constant integral vectors of dimension m ($v_i \in N^m$). The constant vectors v_1,...,v_k are called 'generators' of the linear set, as they are indeed generators of the set S obtained by uniformly subtracting the 'offset vector' v_0 from the linear set L (i.e. $S = \{x - v_0 \mid x \in L\}$), when one views the set as the additive semigroup generated by the v_i's.

The model of learnability we use is that of 'polynomial learnability' (or 'pac-learnability'), introduced by Valiant [18], [17] in the context of boolean concept learning, and subsequently generalized to arbitrary concepts by Blumer et al. [3]. We consider the learnability of semilinear subsets of N^m with respect to both the unary encoding and the binary encoding of integers both for examples and concept representations.

We settle, up to varying degrees of hardness assumptions ranging from $P \neq NP$ and $RP \neq NP$ to the hardness of learning DNF, a host of learnability questions for these and related classes of subsets of N^m. This work provides an interesting class of infinite concepts for which the questions of learnability in Valiant's model have been nearly completely settled, and demonstrates how various proof techniques recently developed by Pitt and Valiant [14], Blumer et al. [3], and Pitt and Warmuth [16] can be fruitfully applied to classes of formal languages.

2 Results and Significance

With respect to the unary encoding, we show that the entire class of semilinear sets, for dimensions 1 and 2, is polynomially learnable. Since m-dimensional semilinear sets are exactly the sets of "letter counts" (or Parikh-images) of regular sets over an m-letter alphabet (see [12]), our result implies that NFA (non-deterministic finite state automata) of alphabet size at most 2 is polynomially learnable *modulo* equivalence under Parikh-mapping, in a certain well-defined sense. This result is particularly interesting, in light of the recent results by Pitt and Warmuth [15] and Kearns and Valiant [10] indicating that learning the entire class of NFA is most likely hard. We show this result by a proof technique due to Blumer et al. [4], namely by exhibiting an 'Occam algorithm', that is, a polynomial time algorithm which reliably compresses a given sample to a consistent hypothesis, which is polynomially approximately minimal and less than linear in the sample size (number of examples).

We complement this result with negative results of varying sorts which, in addition to being of interest on their own, provide specific reasons to suspect that a significantly stronger positive result for this class is difficult to obtain. First, we show that the corresponding 'minimal consistent concept representation' problem (to find a minimal hypothesis consistent with a given sample from the class of concept representations under consideration) is strongly NP-complete. Obtaining an approximately minimal consistent hypothesis, therefore, is the best we can reasonably expect to do in polynomial time. Second, we show that the Vapnik-Chervonenkis dimension[1] of successive subclasses of semilinear sets of bounded size n, where 'size' is taken to mean the size in unary of the concept representations in question, grows at least as fast as $n^{\frac{m}{m+1}}$, hence it follows that the

[1] Vapnik-Chervonenkis dimension, or VC-dimension for short, is a measure of combinatorial complexity of a concept class. A great many of the performance guarantees for polynomial learnability are stated in terms of the VC-dimension of the concept class in question [3].

lower-bound for the number of examples required for pac-learning is also at best order of $n^{\frac{m}{m+1}}$, by a result of Ehrenfeucht et al. [6]. Finally, we show that the 'prediction' problem[2] for linear sets encoded in binary is at least as hard as that for DNF – a long-standing open problem in the field. In fact, the existence of any prediction algorithm for linear sets encoded in *unary*, which runs in time polynomial in the 'domain dimension' (m) in addition to other parameters, would also show the predictability of DNF. Furthermore, the analogous results are shown to hold for a class of subsets of Z^m which is significantly simpler than semilinear sets.

Our result on the minimal consistent concept representation problem for semilinear sets in unary is one of non-approximability. More specifically, it cannot be approximated within any guaranteed constant factor less than 2 in polynomial time, unless $P = NP$. This result complements the recent result of Pitt and Warmuth [15] that the 'minimal consistent DFA problem' cannot be approximated within any polynomial. On the one hand, we do not have the non-polynomial approximability result that they have. On the other hand, the class under consideration here is a much simpler class which is in fact polynomially learnable (if only for a restricted range of dimensions). We also show a related result using a proof technique due originally to Pitt and Valiant [14]; The class of semilinear sets that are unions of a bounded number, say k, of linear sets ('k-fold semilinear sets') is not properly polynomially learnable[3] unless $RP = NP$.

Our results on predictability are shown using the proof technique of 'prediction preserving reducibility' recently developed by Pitt and Warmuth [16]. The result that the class of linear sets encoded in binary is as hard to predict as DNF is not surprising, as we have also shown that even the 'evaluation problem'[4] for linear sets in binary is NP-complete [1]. [5] It was noted by Manfred Warmuth, however, that the prediction-preserving reduction exhibited to prove this result can be extended to show that the prediction problem for linear sets in unary is also as hard as that of DNF, *if* one considers the dimension of the class to be a variable ('variable dimensions'). The class of concepts for which we show the analogous results is the 'submodules' of Z^m, the learning problem of which has been extensively investigated by Helmbold, Sloan and Warmuth [9]. They show that not only is the class of submodules of Z^m encoded in binary efficiently learnable, but so is the class of 'nested differences' of members of this class. [6] In contrast, we have shown that the prediction problem for the class of finite unions of modules (or 'semi-modules') is also as hard as that for DNF, for fixed dimensions in binary, or variable dimensions in unary.

These results together draw up a nearly complete characterization of learnability of the concept classes in question, as summarized in the tables in Figures 1 and 2.[7] The positive results for

[2]The prediction problem for a class of concepts is a slightly relaxed notion of learning where the algorithm need not output any hypothesis concept explicitly. Predictability of a given class is implied by its learnability (cf. [16]).

[3]A class of concept representations $\mathcal{A}$ is said to be learnable *by* another class $\mathcal{B}$, if there exists a learning algorithm which only outputs hypotheses from $\mathcal{B}$ and learns $\mathcal{A}$. $\mathcal{A}$ is *properly* learnable iff $\mathcal{A}$ is learnable by $\mathcal{A}$ (cf. [14]).

[4]The evaluation problem for a class of language representations $\mathcal{G}$ is the language $\{\langle G, w\rangle \mid G \in \mathcal{G} \wedge w \in L(G)\}$ where $L(.)$ denotes the mapping from a representation to the language it represents (cf. [16]).

[5]It has been suggested by several different sources that this result had been known, although no correct reference to the result has been pointed out to the author.

[6]Algebraically these classes are closely related. A linear set is a finitely generated semigroup under addition with a constant offset, while a module is indeed a finitely generated module under addition and subtraction. The former is a much more complicated class of concepts, however, both computationally and learning-theoretically. For the former the evaluation problem (in binary) is NP-complete and the Vapnik-Chervonenkis dimension grows nearly linearly already with respect to the unary encoding, whereas for the latter the evaluation problem is efficiently solvable in polynomial time, and the Vapnik-Chervonenkis dimension grows logarithmically with respect to the unary encoding.

[7]In these tables, 'Thm x.y' indicates that the result is stated in Theorem x.y, and 'Cor x.y' indicates that it follows

	Linear Sets	Semilinear Sets	Modules	Semi-modules
Fixed Dimensions				
unary	yes $m \leq 2$ (Cor 5.2)	yes $m \leq 2$ (Thm 5.2)	yes [9]	yes (Cor 5.2)
binary	$\trianglerighteq$ DNF (Thm 5.5)	$\trianglerighteq$ DNF (Cor 5.5, 5.6)	yes [9]	$\trianglerighteq$ DNF (Thm 5.6)
Variable Dimensions				
unary	$\trianglerighteq$ DNF * (Thm 5.7)	$\trianglerighteq$ DNF * (Cor 5.7, 5.8)	yes [9]	$\trianglerighteq$ DNF (Thm 5.8)
binary	$\trianglerighteq$ DNF (Cor 5.7)	$\trianglerighteq$ DNF (Cor 5.7, 5.8)	yes [9]	$\trianglerighteq$ DNF (Cor 5.8)

Figure 1: Results on pac-learnability and predictability.

Semilinear Sets (Proper Pac-learnability, Minimal Consistency)				
	Linear Sets (proper-pac)	K-fold Semilinear Sets (proper-pac)	Minimal Consistent Semilinear Sets	Semilinear Sets (pac)
unary	?	hard ($k \geq 3$) unless $RP = NP$ (Thm 5.4)	NP-hard (Thm 5.3)	yes ($m \leq 2$) (Thm 5.2)

Figure 2: Results on Semilinear Sets encoded in Unary.

modules are due to Helmbold, Sloan and Warmuth [9], as indicated.

3 The Learnability Models

We are concerned with the question of probably approximately correct(pac) learnability of countably infinite concept classes from randomly generated examples, essentially in the sense of Valiant [18]. In adapting Valiant's original formulation of learnability of boolean concepts to infinitary domains, an interesting question arises as to exactly what parameters quantifying the complexity of the concept class under consideration should be included in the arguments to the polynomial of the 'sample complexity'[8]. In the case of boolean concepts, the sample complexity was allowed to polynomially (and at most polynomially) depend on the number of variables, which could be thought of as either the length of examples (assignments), or the 'dimension' of the domain under consideration. In an infinitary domain such as Σ^*, it is arguable whether the sample complexity should be allowed to polynomially depend on the (maximum) length of examples seen. Subsequent generalizations of Valiant's model to infinitary domains have taken different views on this issue. In this paper we employ different formulations of learnability, depending on whether we are considering the unary encoding or the binary encoding. In particular, we adopt the exact formulation of 'polynomial learnability' by Blumer et al. [3] for the unary case, in which the sample complexity is *not* allowed to depend on the example length. For the binary case, we employ the version of polynomial learnability in which sample complexity is allowed to polynomially depend on the length of the longest example seen, following the formulation of 'predictability' by Haussler et al. [8] and Pitt and Warmuth [16]. It is important to note this distinction particularly because the prediction problem with respect to the unary encoding becomes trivial if the sample complexity is allowed to polynomially depend on the length of the longest example seen [13]. We review in the following

(essentially) as a corollary of Theorem x.y. '$\trianglerighteq$ DNF' indicates that DNF is prediction-preserving reducible to it. '[9]' indicates a result by Helmbold et al., and '*' an extension by Manfred Warmuth.

[8]This notion will shortly be formally defined.

the definition of 'polynomial learnability'.

In this paper the domain (X) under consideration is assumed to be $(\Sigma^*)^m$ for some alphabet Σ. Learnability question is asked of a class of *concept representations* ($\mathcal{G}$) which represent some class of concepts $R(\mathcal{G})$ over X, that is[9], $R(\mathcal{G}) \subseteq \mathcal{P}(X)$. We fix some encoding scheme for these representations with respect to which we define the notion of 'complexity measure' (*size*) on these representations. We let $\mathcal{G}_s$ denote the subclass of $\mathcal{G}$ of bounded size s. Examples are drawn with respect to some time-invariant probability distribution (D) over the entire domain X, and then they are paired with either $+$ or $-$ to indicate whether they belong to the target concept or not. This process is iterated m times to obtain a *labeled sample* of size m (S_m), which is then fed into our learning algorithm.

A learning algorithm is a possibly randomized algorithm which takes as input a labeled sample and outputs a hypothesis from some class of concept representations. A learning algorithm is said to *learn* a concept representation class $\mathcal{G}$ with sample complexity $f(\epsilon, \delta, s)$ if and only if for whatever distribution D, and whatever values of ϵ and δ, whenever it is fed with a randomly generated labeled sample of size at least $f(\epsilon, \delta, size(G))$ for some concept $R(G)$ in $R(\mathcal{G})$ according to D, its output hypothesis H is an ϵ-approximation of $R(G)$, that is, $D(R(G) \triangle R(H)) \leq \epsilon$, with probability at least $1 - \delta$. A class of concept representations $\{\mathcal{G}_s \mid s \in N^+\}$ is said to be *polynomially learnable*, if there exists a learning algorithm A which learns $\mathcal{G}$ with sample complexity $f(\epsilon, \delta, s)$, for some f polynomial in ϵ^{-1}, δ^{-1}, and s, and which runs in time polynomial in the total length of the input sample.

For the binary case, we use the version of polynomial learnability in which on any particular 'run', the distribution (D) is assumed to be defined over a subset of the domain Σ^* of bounded length n, and then a learning algorithm is said to polynomially learn a class of concept representations just in case it learns it with sample complexity $f(\epsilon, \delta, s, n)$ for some f polynomial in all of ϵ^{-1}, δ^{-1}, s and n.

4 Classes of Concepts Under Consideration

We define the class of concept representations considered in this paper. Throughout, $b_0, .., b_n$ are integral vectors in N^m.

Linear Bases (LB) for Linear Sets
$B = \langle b_0, \{b_1, ..., b_n\}\rangle$ is called the *linear basis* of the linear set $L(B) = \{b_0 + \sum_{i=1}^{n} x_i b_i \mid \forall i \leq n\ x_i \in N\}$.

Semilinear Basis Sets (SLB) for Semilinear Sets
If $B_1, ..., B_n$ are linear bases, then $\{B_1, ..., B_n\}$ is called the *semilinear basis set* of the set $\cup_{i=1}^{n} L(B_i)$.

Module Bases (MB) for Modules
The set $B = \{b_1, ..., b_n\}$ is called the *module basis* of the module $L(B) = \{\sum_{i=1}^{n} x_i b_i \mid \forall i \leq n\ x_i \in Z\}$.

[9] We denote by $R(G)$ the concept represented by G, and by $R(\mathcal{G})$ the *class* of concepts represented by $\mathcal{G}$.

Semi-module Basis Sets (SMB) for Finite Unions of Modules
If $B_1, ..., B_n$ are module bases, then $\{B_1, ..., B_n\}$ is called the *module basis set* of the set $\cup_{i=1}^{n} L(B_i)$.

The components of example vectors in the domain, as well as those of vectors in concept representations are either encoded in binary, or encoded in unary. We define 'size' in binary of a concept representation to be the sum of the log of all the integers appearing as components of vectors in it, and the 'size' in unary to be the sum of all the component integers themselves. In this paper we consider the two cases (out of the four in total), in which the same encoding is used for the examples and for the concept representations. We explicitly quantify all our results by 'unary' or 'binary', as in '$SLB(unary)$', for example. Also we specify the dimension of the concept class under consideration in a similar manner: For example, $SLB(m, binary)$ denotes the m-dimensional SLB in binary, and $LB(variable, unary)$ denotes variable-dimensional LB in unary. Finally, for any concept representation class A (such as SLB) we denote by A_n (such as SLB_n) the subclass of A of bounded size n, that is, $A_n = \{B \in A \mid size(B) \leq n\}$.

5 Technical Details

5.1 Vapnik-Chervonenkis Dimensions of Semilinear Sets in Unary

We characterize the VC-dimension of LB_n of dimension 1 up to a constant factor, and those of LB_n and SLB_n of arbitrary dimensions within at most $\sqrt{n}$ factor, and thereby establish a lowerbound for the sample complexity of any learning algorithm for these classes by a result of Ehrenfeucht et al. [6].

Theorem 5.1 *If we let $VCdim(C)$ denote the VC-dimension of the concept class represented by C,*

(1) $\forall n \in N^+ \; VCdim(LB(1, unary)_n) = \Theta(\sqrt{n})$
(2) $\forall n \in N^+ \; VCdim(LB(m, unary)_n) = \Omega(\sqrt{n})$
$= O(n)$

(3) $\forall m \in N^+ \; \forall n \in N \; VCdim(SLB(m, unary)_n) = \Omega(n^{\frac{m}{m+1}})$
$= O(n)$

Corollary 5.1 *Any learning algorithm for SLB(m, unary) requires at least $\Omega(\epsilon^{-1} \cdot \log \delta^{-1} + \epsilon^{-1} \cdot n^{\frac{m}{m+1}})$ many examples to achieve ϵ accuracy with $1 - \delta$ confidence.*

In both cases, the upperbound of $O(n)$ trivially follows from the fact that a string of length n can represent at most $O(2^n)$ different concepts.

Proof of Theorem 5.1(1)
Lowerbound: Let $S_n = \{n, n+1, ..., 2n-1\}$. Then, for an arbitrary subset $T \subseteq S_n$, if we let $B_T = \langle 0, T \rangle$, then, $L(B_T) \cap S_n = T$. Also, $size(T) = \sum_{x \in T} x \leq \sum_{x \in S_n} x \leq 2 \cdot n^2$. Hence, S_n is shattered by $LB(1, unary)_{2n^2}$ for an arbitrary n. It follows therefore that for some constant c, $\forall n \in N \; VCdim(LB(1, unary)_n) \geq c \cdot \sqrt{n}$.

Upperbound: This is shown by a counting argument. If we let $P(n)$ denote the number of 'partitions' of n, that is, the number of multisets of positive integers whose sum equal n, then we have $card(LB(1, unary)_n) = O(\sum_{i=1}^{n} P(n))$. By a result by Hardy and Ramanujan (See for example [2].), we have $P(n) = \Theta(2^{\sqrt{n}})$. So, $\sum_{i=1}^{n} P(i) = \Theta(\sum_{i=1}^{n} 2^{\sqrt{i}}) = \Theta(2^{\sqrt{n}})$. Hence, $VCdim(LB(1, unary)_n) \leq \log(card(LB(1, unary)_n)) = \Theta(\sqrt{n})$.

Proof of Theorem 5.1(3)
We prove the lowerbound. Let $S_{n,m} = \{1, 2, .., n^{\frac{1}{m}}\}^m$, where we assume that n is a perfect m-th power. Note the following:[10]
(1) $card(S_{n,m}) = n$.
(2) $size(S_{n,m}) = m \cdot n^{\frac{m-1}{m}} \cdot \sum_{i=1}^{n^{\frac{1}{m}}} i = O(m \cdot n^{\frac{m-1}{m}} \cdot n^{\frac{2}{m}}) = O(n^{\frac{m+1}{m}})$.
Now, for an arbitrary $T \subseteq S_{n,m}$, define $\mathcal{B}_T = \{\langle b, \phi \rangle \mid b \in T\}$. Then for any T, it is clear that $L(\mathcal{B}_T) \cap S_{n,m} = T$, and $size(\mathcal{B}_T) \leq size(S_{n,m}) = O(n^{\frac{m+1}{m}})$. thus, $S_{n,m}$ is shattered by $SLB(m, unary)_{n^{\frac{m+1}{m}}}$: Hence, $VCdim(LB(m, unary)_n) = \Omega(n^{\frac{m}{m+1}})$.

5.2 Learnability Result for Semilinear Sets

Theorem 5.2 *SLB(m, unary) is Properly Polynomially Learnable for m = 1, 2.*

We prove this theorem (for the 2 dimensional case only) by exhibiting an Occam Algorithm with a range dimension (cf. [4]) logarithmic in the sample size, and polynomial in the size of a minimal basis set.

5.2.1 Proof Sketch of Theorem 5.2

We exhibit a normal form for semilinear bases, with a 'polynomial blow-up',[11] such that each semilinear basis set in this normal form is a finite union of linear bases each of which has at most two generators. Now, for a given positive sample, we can generate all such linear bases that are *relevant* to that sample and eliminate the ones that are inconsistent with the negative sample in time polynomial in the total sample length. This leaves us with a polynomially bounded set of linear bases of *varying* sizes all consistent with the input sample, of which the 'relevant' part of a minimal consistent semilinear basis set in the normal form is guaranteed to be a subset. We can then apply the polynomial time approximation algorithm of Chvatal [5] to the instance of Weighted Set Cover obtained by taking the positive sample to be the set to be covered, its subsets defined by the relevant linear bases and the sizes of these bases to be the weighted legal subsets. As Chvatal's algorithm always outputs a cover whose total weight is polynomially bounded in the total weight of a minimal weight cover and logarithmic in the number of points to be covered, this will give us an Occam algorithm.

The main significance of this result lies in the fact that via the 'poly-blowup' normal form, we are able to show the entire class of semilinear sets of dimensions 1 and 2 to be learnable, without

[10] For any set S of vectors, we define $size(S)$ to be the sum of all the components of vectors in it.
[11] That is, for each semilinear basis set there exists a language equivalent one in the normal form whose size is only polynomially larger, for some fixed polynomial.

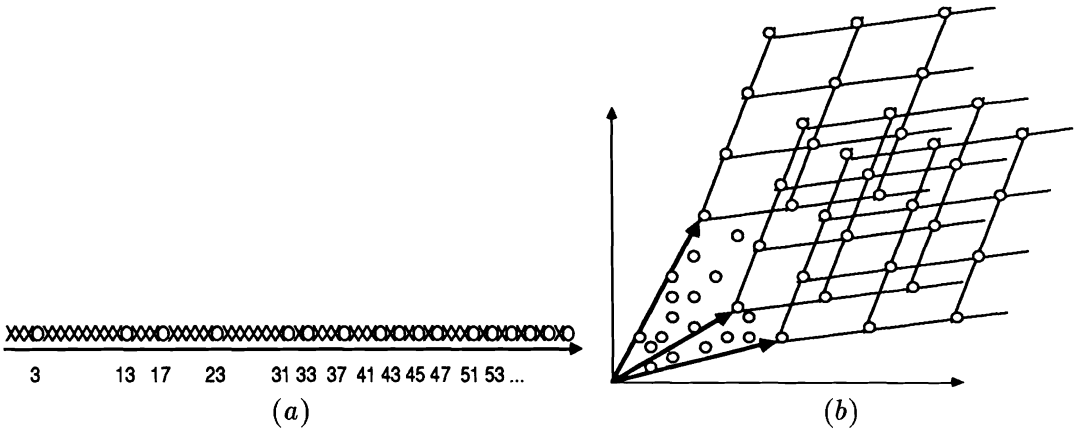

Figure 3: Schematic views of 1,2-dimensional linear sets.

having to put an explicit bound on the number of generators, as is necessary, for example, in the case of k-DNF. It is worth noting that our proof in some sense uses a special case of the notion of 'prediction preserving reduction'. Namely, we have reduced the learning problem of unrestricted SLB to that of '2-SLB', by a reduction with the concept mapping being the 'poly-blowup' normal form, and the example mapping being the identity function.

5.2.2 The Normal Form Lemma

As it is clear from the proof sketch we just gave, the key step in the proof is the poly-blowup normal form lemma. We now state the normal form theorem for (2-dimensional) semilinear bases formally.

Lemma 5.1 *There is a polynomial p such that for every semilinear basis set $\mathcal{B}_1$ in SLB(2), there is another semilinear basis set $\mathcal{B}_2$ such that*

(1) $size(\mathcal{B}_2) \leq p(size(\mathcal{B}_1))$
(2) $\forall B \in \mathcal{B}_2\ card(generators(B)) = 2$
(3) $L(\mathcal{B}_1) = L(\mathcal{B}_2)$

The intuitive reason why the above holds is that a linear set is ultimately periodic. For analogy let us first consider the 1-dimensional case. A 1-dimensional linear set looks very complicated close to 0. However, past a certain point called the conductor [11], it becomes very simple, i.e. an integer is in the set if and only if it is expressible as the sum of the offset and some multiple of the greatest common divisor of all the generators of its basis. (See Figure 3(a).) Furthermore, the conductor of any linear set is polynomially bounded in the size of its basis.

In the 2-dimensional case, the exact analogue of the 'conducting' phenomenon does not happen. A slightly weaker phenomenon does happen, however, which suffices for our present purpose. A 2-dimensional linear set is also complicated around the origin (0,0), but if one goes sufficiently (polynomially) far from it, then a point is in the set if and only if it is expressible as the sum of a linear combination of some *fixed* pair of vectors and one of a polynomially bounded set of polynomially small offset vectors. In other words, each linear set is equivalent to a semilinear set

consisting of polynomially many linear sets, each of which has a basis with two generator vectors. Hence, so does each semilinear set. (See Figure 3(b).)

The first key fact for showing this is the following. Given an arbitrary pair of vectors, say v_1 and v_2, and any vector, say v, in between[12] them, there is a multiple of v, which is at most the determiner of the matrix $A = [v_1, v_2]$, which is expressible as a linear integral combination of v_1 and v_2. The determiner of A is, in turn, bounded by a fixed polynomial a^m where a is the maximum component in A – a quadratic function for the case $m = 2$. Thus, given an arbitrary set of generator vectors, the two vectors that are 'rightmost' and 'leftmost' among them can express every sufficiently large multiple of every other vector in the set as their linear combination.

Given this fact, it is easy to see that the normal form lemma holds. Given an arbitrary set of *two-dimensional* generator vectors, say $V = \{v_1, ..., v_n\}$, one can always pick the 'rightmost' one and the 'leftmost' one, say v_1 and v_2.[13] Then any linear combination w of vectors in V is expressible as $\Sigma_{i=1}^{n} c_i \cdot v_i$, where all of integral constants c_i's, except for c_1 and c_2, are less than or equal to some integer, say d, which is at most order of $(size(v))^2$. Hence given an arbitrary linear basis $B = \langle v_0, \{v_1, ..., v_k\}\rangle \in LB(2)$ (where we assume without loss of generality that v_1 is the "rightmost" one, and v_2 is the "leftmost" one among v_1 through v_k), if we take the set of linear bases $\mathcal{B} = \{\langle (v_0 + y_3 \cdot v_3 + ... + y_k \cdot v_k), \{v_1, v_2\}\rangle \mid y_3, ..., y_k \leq d\}$ then $L(B) = L(\mathcal{B})$. Furthermore, if we let $n = size(B)$, then the size of $\mathcal{B}$ is seen to of order n^9, because each of the offset vectors in $\mathcal{B}$ has size order of n^3, and there are at most $o(n^6)$ many different offsets of the form $v_0 + y_3 \cdot v_3 + ... + y_k \cdot v_k$. That the size of each offset is of order n^3 follows because each multiple y_i is of order n^2, and there are k offsets: Clearly $k \leq n$. To obtain the bound on the number of distinct offsets, we observe that each offset is a 2-dimensional vector with both of its coordinates bounded by n^3, and hence there can be n^6 such vectors at the most. Hence $\mathcal{B}$ satisfies all the conditions of Theorem 5.1.

5.3 NP-completeness of the Minimal Consistent Concept Problem

The results in this section concern the problem of finding a consistent concept (representation) for a given labeled sample, satisfying a certain minimality constraint. We define this as an optimization problem below, and state our results in terms of it.

MCC($\mathcal{G}, measure$)
for a class $\mathcal{G}$ of concept representations with associated measure *measure* on them to be minimized;
INSTANCE: A finite labeled sample S.
PROBLEM: Find G in $\mathcal{G}$ which is consistent with S, and $measure(G) = min\{measure(F) \mid consistent(F, S)\}$.

Theorem 5.3 *MCC(SLB(unary), size) cannot be approximated within any constant less than 2 in polynomial time unless $P = NP$.*

[12] In other words, cosine of v is in between those of v_1 and v_2.

[13] The corresponding statement for the 3-dimensional case is false, and this is why the above lemma in the 3-dimensional case does not give rise to the analogue of the next lemma. The same technique does not yield an Occam Algorithm for the 3-dimensional and higher dimensional cases.

Theorem 5.4 *MCC(SLB(unary), cardinality) cannot be approximated within any constant less than 2 in polynomial time unless $P = NP$.*

By a lemma (Lemma 5.2) which is essentially due to Pitt and Valiant [14], our proof of Theorem 5.4 implies that the subclasses of SLB with a bounded 'cardinality' (k-fold-SLB) are not properly polynomially learnable (for $k \geq 3$), provided that $RP \neq NP$.

Lemma 5.2 *Let A be a class of concept representations, and L an NP-hard language. If there exists a polynomial time transformation τ from L to finite samples and a polynomial p such that all of the following conditions are equivalent, then A is not properly polynomially learnable, unless $RP = NP$.*

(1) $x \in L$
(2) There exists $G \in A$ which is consistent with $\tau(x)$.
(3) There exists $G \in A$ which is consistent with $\tau(x)$, and $size(G) \leq p(size(x))$.

Corollary 5.2 *k-fold-SLB(unary) is not properly polynomially learnable for $k \geq 3$, unless $RP = NP$.*

We give a sketch of the proof of Theorem 5.4, and then explain how it can be modified to prove Theorem 5.3.

5.3.1 Proof Sketch of Theorem 5.4

We exhibit a polynomial time transformation from instances of Graph-k-Colorability (GkC) [7] to finite samples such that there exists a (1-dimensional) k-fold semilinear set consistent with the resulting sample, just in case the original graph is k-colorable. We give a rough outline of this transformation.

Given an arbitrary graph, we 'represent' each vertex in the graph by a unique integer (call them 'vertex numbers') and put them in the positive sample. We then put, for each edge in the graph, the sum of the numbers representing the two end vertices of the edge in the negative sample. Further, by means of additional negative examples, we enforce that any linear basis that generates a vertex number must include that very number either in its generators or as its offset, and hence if it generates any two vertex numbers, it must also generate their sum. First, we add all the integers between 0 and the maximum vertex number, which have not already been put into the sample, to our negative sample. We then add for each pair of vertex numbers, say $a < b$, the integer $a+2(b-a)$ into our negative sample thereby ensuring that no vertex number could be generated as a linear combination of another vertex number and the difference between the two.

The Transformation
$G = \langle V, E \rangle$ where $V = \{v_i \mid 1 \leq i \leq n\}$ and $E \subseteq \{(v_i, v_j) \mid v_i, v_j \in V\}$ is transformed to $S = S^+ \cup S^-$, where T_n is the set of n vertex numbers to be specified later (in Lemma 5.4).

$$
\begin{aligned}
S^+ &= \{\langle t_i, + \rangle \mid t_i \in T_n\} \\
S^- &= \{\langle t_i + t_j, - \rangle \mid (v_i, v_j) \in E\} \\
&\cup \{\langle x, - \rangle \mid 0 < x < max(T_n) \ \wedge \ x \notin T_n\} \\
&\cup \{\langle t_j + (t_j - t_i), - \rangle \mid i, j \leq n \ \& \ t_i < t_j\}
\end{aligned}
$$

Since we make sure that each vertex number must be 'colored', but no two vertex numbers are

to be 'colored' by the same linear set if there is an edge between them, the resulting sample should have a consistent k-fold-semilinear set if and only if the original graph is k-colorable. Note further that when there is a k-fold semilinear basis set consistent with a sample generated in the above manner, there is one that has as its offsets and periods all and only the vertex numbers of the transformation. Thus the size of a minimal consistent semilinear basis set is predictably small. We summarize this in the following lemma, which by Lemma 5.2 implies Corollary 5.2.

Lemma 5.3 *If G is any graph of n vertices, and S is the sample that the above transformation maps G to, then all of the following conditions are equivalent.*

(1) G is k-colorable.
(2) There exists a k-fold-semilinear basis set consistent with S.
(3) There exists a k-fold-semilinear basis set consistent with S, and is of size $\sum_{i=1}^{n} t_i$.

In order to verify Lemma 5.3, we must demonstrate that the outlined transformation can be carried out without either accidentally 'putting an edge' where there is none, or making the resulting sample inconsistent.[14] We specify a certain set of conditions for the vertex numbers (T_n) which suffice for this purpose, and show that for an arbitrary number of vertices (n), a set of vertex numbers T_n satisfying such conditions can be quickly computed. This is formalized in the following lemma.

Lemma 5.4 *There is an algorithm, which on input $n \in N$, computes in time polynomial in n, a set T_n of n integers with the following properties.*

1. $\sum_{x \in T_n} x \leq q(n)$ *for a fixed polynomial q.*
2. (a) $\forall u, v, w, x \in T_n\ [(\{u, v\} \neq \{w, x\}) \rightarrow (u + v \neq w + x)]$
   (b) $\forall x, y, z \in T_n\ [x + y > z]$
3. (a) $\forall x, y, z \in T_n [(x < y) \rightarrow (y + (y - x) \neq z)]$
   (b) $\forall x, y, z_1, z_2 \in T_n [(x < y) \rightarrow (y + (y - x) < z_1 + z_2)]$

Proof Sketch of Lemma 5.4
First note that *3(a)* in fact follows from *2(a)*, because if there were $x, y, z \in T_n$ such that $y+(y-x) = z$, then we would have $x + z = y + y$, which contradicts *2(a)*. Note also that *2(b)* follows from *3(b)*. Thus, we need only be concerned with *1*, *2(a)*, and *3(b)*. Furthermore, if we can show that a set of n integers, say S_n, with properties *1* and *2(a)* can be generated in polynomial time, then we can obtain T_n by adding $2 \cdot max(S_n) + 1$ to each member of S_n so that *3(b)* is satisfied. We can do this because property *2(a)* is preserved under any "translation" of the set by a constant offset.

We are left to verify that S_n can indeed be computed from n in polynomial time. We define $S_n = \{s_1, ..., s_n\}$ in stages (iterations).

Stage 1
Let $S_1 = \{0\}$ and $S_1^2 = \{0\}$.
Stage i
Let $S_i = \{s_i\} \cup S_{i-1}$ where $s_i = min\{x \in N \mid x > max(S_{i-1}) \wedge (\forall y \in S_{i-1} \forall z \in S_{i-1}^2\ [x + y \neq z])\}$.
Let $S_i^2 = \{s_i + s_j \mid s_i, s_j \in S_i\}$.

It is easy to see that for each n, S_n satisfies property *2(a)*. For suppose otherwise, then for

[14] That is, the same integer is never put both to the positive and negative sample.

some $s_i, s_j, s_k, s_l \in S_n$ such that $\{s_i, s_j\} \neq \{s_k, s_l\}$ we have $s_i + s_j = s_k + s_l$. Pick the maximum index among i, j, k and l, say l. Then all of i, j, k are strictly less than l, for if $i = l$ then, that would imply $j = k$ and the two sets are identical, and if $k = l$ then we would have to have $i = j = k = l$ for the equality to hold because s_l is maximal. Thus, at stage l, $s_k \in S_{l-1}$, and $s_i + s_j \in S^2_{l-1}$. So, letting $y = s_k$ and $z = s_i + s_j$ in the minimization clause, this would have rejected s_l as an x satisfying the condition. This is a contradiction. It is straightforward to verify that all the members of S_n are bounded by a fixed polynomial in n (in fact n^4), and that S_n can be computed in time polynomial in n (at most $O(n^4 \cdot \log n)$).

5.3.2 Proof Sketch of Theorem 5.3

We modify the above transformation to prove Theorem 5.3 as follows: We add to every example in the sample some constant offset $a > max(T_n)$, chosen depending on the constant of non-approximability and n, add a in the positive sample and all the integers less than a in the negative sample. More precisely, the new sample S_a is defined as $S_a = S_a^+ \cup S_a^-$ where:

$$S_a^+ = \{\langle x + a, + \rangle \mid \langle x, + \rangle \in S^+\} \cup \{\langle a, + \rangle\}$$

$$S_a^- = \cup\{\langle x + a, - \rangle \mid \langle x, - \rangle \in S^-\} \cup \{\langle 2a, - \rangle\} \cup \{\langle x, - \rangle \mid x \in N \wedge 0 \leq x < a\}$$

First we note that if some linear basis B consistent with S_a generates $\{a + x \mid x \in A\}$ for some subset A of T_n then we must have either of the following two cases.
(i) B's offset equals a. Its generators must contain A. We say that such a B is of 'type 1' and write $type(B, 1)$.
(ii) B's offset does not equal a. If its offset is 0 then its generators contain $\{a + x \mid x \in A\}$ because it cannot contain a as a generator to respect $\langle 2a, - \rangle$ in S_a^-. Otherwise, A must be a singleton, and B's offset must equal that one member in A. We say that such a B is of 'type 2', and write $type(B, 2)$.

The crucial fact is that since all the required properties of T_n of Lemma 5.4 are preserved under positive 'translation', by a in this case, essentially the same argument as before applies, *if* all of the linear bases in question are of type 1: Namely, there is a k-fold-SLB all of whose bases are of type 1 consistent with S_a if and only if the original graph G is k-colorable. Using this fact, we can verify the following 'gap' lemma.

Lemma 5.5 *Let S_a be the sample obtained as above from an arbitrary graph G, and $\mathcal{B}$ a minimal consistent SLB for it. Then we have:*

(1) If G is k-colorable, then $size(\mathcal{B}) = (\sum_{x \in T_n} x) + k \cdot a$.
(2) If G is not k-colorable, then $size(\mathcal{B}) \geq (\sum_{x \in T_n} x) + (k + 1) \cdot a$.

Proof of Lemma 5.5
If the graph G is k-colorable, then there is a k-fold semilinear basis set consistent with S_a, such that each of its linear basis is of type 1, and hence has size $k \cdot a + \sum_{i=1}^{n} t_i$. Suppose on the other hand that G is not k-colorable, and let $\mathcal{B} = \{B_i \mid i = 1, ..., l\}$ be a minimal consistent SLB for S_a

and let $Q_i = L(B_i) \cap \{a + t_i \mid t_i \in T_n\}$. Now let $I = \{i \leq l \mid type(B_i, 1)\}$, $J = \{i \leq l \mid type(B_i, 2)\}$, and $R = \cup_{i \in J} Q_i$. Then, note that:

$$size(\mathcal{B}) \geq \sum_{x \in T_n} x + a \cdot (card(I) + card(R))$$

We claim that we must have $card(I) + card(R) \geq k + 1$. For suppose otherwise, i.e. $card(I) + card(R) \leq k$. Then define $\mathcal{B}' = \{B_i \mid i \in I\} \cup \{\langle a, \{x - a\}\rangle \mid x \in R\}$. Note that (i) all of the bases in $\mathcal{B}'$ are of type 1, (ii) $card(\mathcal{B}') \leq k$, and (iii) $\mathcal{B}'$ is consistent with S_a. Hence it follows that G must be k-colorable, contradicting our hypothesis. Thus, we have shown that $card(I) + card(R) \geq k + 1$, and hence $size(\mathcal{B}) \geq \sum_{x \in T_n} x + a \cdot (k + 1)$. □

By appropriately setting a as a polynomial function of $\sum_{i=1}^{n} t_i$ and ϵ^{-1}, we can show that any approximation algorithm for MCC($SLB(unary), size$) with a guaranteed constant factor $2 - \epsilon$ can be used to approximate GkC within $2 - \frac{\epsilon}{2}$, which is known to be NP-hard [7].

5.4 Prediction Preserving Reductions from DNF

The results in this section are all stated in terms of the notion of 'prediction preserving reducibility' due to Pitt and Warmuth [16]. We note that if a class of concepts A is prediction-preserving reducible to B (written $A \trianglelefteq B$), then the predictability of B implies that of A.

Theorem 5.5 $\forall m \in N^+$ *DNF* $\trianglelefteq$ *LB(m,binary).*

Theorem 5.6 $\forall m \in N^+$ *DNF* $\trianglelefteq$ *SMB(m,binary), and DNF* $\trianglelefteq$ *SLB(m,binary).*

Theorem 5.7 *DNF* $\trianglelefteq$ *LB(variable, unary).*

Theorem 5.8 *DNF* $\trianglelefteq$ *SMB(variable, unary), and DNF* $\trianglelefteq$ *SLB(variable, unary).*

To show that DNF $\trianglelefteq$ R, for a class of representations for concepts over N , we must exhibit the following two mappings [16] : the 'example mapping' $f : \{0,1\}^* \times N \times N \rightarrow N$, mapping any assignment to an integer, and the 'concept mapping' $g : DNF \times N \rightarrow R$ mapping any DNF-formula A to a semilinear basis set, satisfying the following conditions.[15]
(1) $\forall s, n \in N\ \forall w \in \{0,1\}^n\ \forall A \in DNF^{[s]}\ f(w, s, n) \in L(g(A, n))$ if and only if w satisfies A.
(2) f is computable in time polynomial in n and s.
(3) g is 'poly-blowup', that is, for some fixed polynomial q, $\forall n, s \in N\ \forall A \in DNF^{[s]}\ size(g(A, n)) \leq q(n, s)$.

In each of the reductions to be exhibited in the sections to follow, no use is made of the variable 'offset' (available for LB and SLB only), i.e. it is always the zero vector. We therefore abbreviate the linear basis $\langle \vec{0}, B \rangle$ by the set of generators B for readability, in the expositions below.

[15] $DNF^{[s]}$ denotes the subclass of DNF with at most s terms.

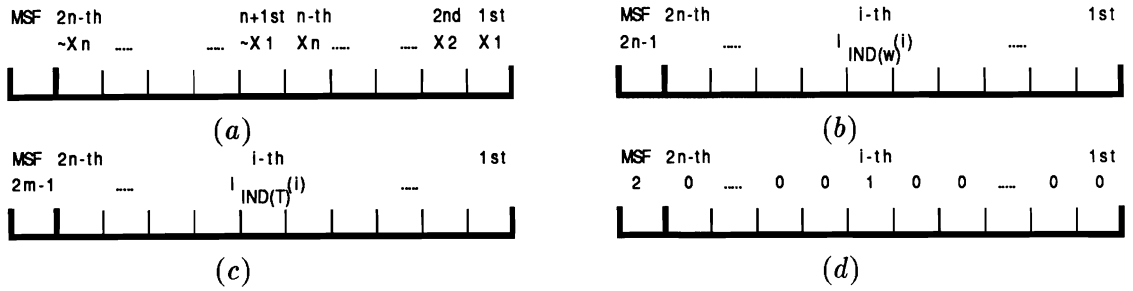

Figure 4: Bit-maps for $f(w,s,n)$, $g'(T,n)$, and e_i.

5.4.1 Proof Sketch of Theorem 5.5

We use the idea of 'bit maps' in our transformation. The integers that are yielded by the example mapping or the concept mapping all have a bit-map representation of the form in Figure 4(a). This map has $2n$ fields each of $q(n)$-bits, where q is some polynomial, plus the most significant field (MSF). The $2n$ fields are to correspond to the $2n$ literals, say; X_1, ..., X_n, $\neg X_1$, ..., $\neg X_n$, in that order from right. We make use of the following notation: If w is an assignment of n variables: $IND(w) = \{i \mid w_i = 1\} \cup \{n+i \mid w_i = 0\}$. If T is a term, then $IND(T) = \{i \mid X_i \in T\} \cup \{n+i \mid \neg X_i \in T\}$. $I_{IND(x)}$ denotes the characteristic function for $IND(x)$, i.e. $I_{IND(x)}(i) = 1$ if $i \in IND(x)$ and 0 otherwise.

f maps an assignment w to an integer whose bit representation is as in Figure 4 (b); It contains $I_{IND(w)}$ in its first $2n$ fields, and $2n-1$ in its most significant field. g maps a DNF A to the union of the set $\{g'(T) \mid T \in A\}$, and the 'extra' numbers, $E(n) = \{e_1, ..., e_{2n}\}$, which will serve the role of 'stuffings'. g' maps a term T to an integer whose bit representation is as in Figure 4 (c) ; Each i-th field contains $I_{IND(T)}(i)$, and the MSF is $2m-1$, where m is the number of literals in T. The 'extra' generators also have the same format: $E(n)$ is the set of $2n$ integers e_1,...,e_{2n} where each e_i has the bit representation in Figure 4 (d); e_i has 2 in its MSF, and 0's everywhere except in the i-th field where it has 1.

The claim is that $f(w,s,n)$ is generated by $g(A,n)$ just in case w satisfies A. We give a brief, informal explanation of why this claim holds. The first crucial fact is that $IND(T) \subseteq IND(w)$ if and only if $w \models T$, for any term T. The next crucial fact is that *if* any linear combination of $g(A,n)$ generates $f(w,s,n)$, then there can be no 'carries'. Therefore, if for any term T, $g'(T)$ is in some linear combination generating $f(w,s,n)$, then T must be satisfied by w. Finally, the fact that MSF's of $f(w,s,n), g'(T)$ and e_i are $2n-1$, $2m-1$, and 2, respectively, ensures that if any linear combination is to equal $f(w,s,n)$, then it must contain a non-zero multiple of $g'(T)$ for some term T. This T must be satisfied by w. It is easily seen, on the other hand, that if there is a term T in A that is satisfied by w, then the sum of $g'(T)$ and the appropriate stuffings; $\{e_i \mid i \in IND(w) \setminus IND(T)\}$, equals $f(w,s,n)$.

It is easy to check that f can be computed in time polynomial in n and s, and that g is 'poly-blowup'.

5.4.2 Proof Sketch of Theorem 5.6

First, we note the following fact, which is essentially a corollary to the Prime Number Theorem.[16]

Fact 5.1 *There exists an algorithm which takes an integer n as input and outputs $2n$ distinct primes in time polynomial in n. We let h denote the function computed by one such algorithm, and let $h(n)$ denote the output of h on n, and $h(n,i)$ the i-th smallest element in $h(n)$.*

These $2n$ primes are then associated with the $2n$ literals there are for n variables: We map any assignment w to the product of the associated primes for those n literals made true by the assignment, say $f(w)$. (For simplicity, we ommit other parameters to f for now.) We then map any term T to the product of the associated primes for all the literals in it, $g'(T)$. The simple but crucial observation is that $g'(T)$ divides $f(w)$ if and only if T is a subset of the set of literals made true by w. Thus, $f(w)$ is in the linear set (module) generated by $\{g'(T)\}$, if and only if w satisfies T. It follows immediately then, that if A is a DNF formula, then $f(w)$ is in the semilinear set (semi-module) generated by $g(A) = \{\{g'(T)\} \mid T \in A\}$ if and only if w satisfies A.

We formally define f and g (for SLB only). Since f is polynomial time computable by Fact 5.1 and g is easily seen to be poly-blowup, the foregoing informal argument shows that they satisfy all the required conditions of a prediction-preserving reduction of DNF to either SMB or SLB.

$$f(w,s,n) = \prod_{i \in IND(w)} h(n,i)$$

$$g(A,n) = \{g'(A_j,n) \mid A_j \in A\}$$

where

$$g'(T,n) = \{\prod_{i \in IND(T)} h(n,i)\}$$

5.4.3 Proof Sketch of Theorem 5.7

The reduction is identical to the one in the proof of Theorem 5.5, except for the fact that the $2n$ fields and MSF of bit maps in the previous case are replaced by $2n$ independent dimensions. Namely, we define our f and g as follows.

$$f(w,s,n) = \langle 2n-1, I_{IND(w)}(2n), ..., I_{IND(w)}(i), ..., I_{IND(w)}(1)\rangle$$

$$g(A,n) = \{g'(T) \mid T \in A\} \cup E(n)$$

where

$$g'(T) = \langle 2m-1, I_{IND(T)}(2n), ..., I_{IND(T)}(i), ..., I_{IND(T)}(1)\rangle$$

$$E(n) = \{e_i \mid 1 \leq i \leq 2n\}$$

$$e_i = \langle 2, 0, ..., 0, 1, 0, ..., 0\rangle$$

An essentially identical argument as before shows that this gives us a prediction preserving reduction from DNF to LB(variable-dimension, unary).

[16] The prime number theorem states that the number of primes less than or equal to n is of order $\frac{n}{\log n}$ – in fact $\Theta(\frac{n}{\log n})$. This, together with the fact that primality checking is performable in pseudo polynomial time, implies the claimed fact.

5.4.4 Proof Sketch of Theorem 5.8

The reduction is similar to the one in the proof of Theorem 5.6 with some twist. Here, instead of the $2n$ primes that were associated with the $2n$ literals for n variables in the previous case, we use $2n$-dimensional unit vectors for the same purpose. In essence, we use the variable dimension at hand to express $2n$ independent components which, in the previous case, we used primes for.

We map any assignment w to the sum of the unit vectors associated with exactly those n literals made *false* by the assignment, and call this $f(w)$. (Again for simplicity, we ommit other parameters to f for now.) Recall that, in the previous reduction, we mapped w to the product of all the primes for those literals made *true* by the assignment. We then map any term T to the basis (with 0 offset) and the set of generators consisting of the associated unit vectors for all the literals *not* in it, denoted $g'(T)$. The crucial fact is that $g'(T)$ can generate $f(w)$ as an integral linear combination of its elments if and only if T contains no literals that are made false by w. Thus, $f(w)$ is in the linear set (module) generated by $g'(T)$ if and only if w satisfies T. Hence if A is a DNF formula, then $f(w)$ is in the semilinear set (semi-module) generated by $\{g'(T) \mid T \in A\}$ if and only if w satisfies at least one of the terms in A.

We formally define the mappings f and g in the following (for SLB only). We introduce the notation '$NIND(.)$' as a short hand for the 'complement' of $IND(.)$, i.e. if w is an assignment then $NIND(w) = \{i \mid w_i = 0\} \cup \{n + i \mid w_i = 1\}$.

$$f(w, s, n) = \langle I_{NIND(w)}(2n), ..., I_{NIND(w)}(i), ..., I_{NIND(w)}(1) \rangle$$

$$g(A, n) = \{g'(T, n) \mid T \in A\}$$

where

$$g'(T, n) = \{e_i \mid i \in NIND(T)\}$$

and e_i is the unit vector whose only non-zero component is its i-th component.

6 Open Problems

Our characterization of learnability is complete, up to various degrees of hardness assumptions, except the learnability question for semilinear sets in unary for dimensions 3 and higher which is an open problem. Also, the question of proper-learnability of linear sets for any dimension is open, though they are clearly learnable by semilinear sets for dimensions up to 2.

Acknowledgments

The author gratefully acknowledges his advisor, Scott Weinstein, for guiding him to the learnability questions of semilinear sets. Thanks are also due to Herbert S. Wilf for his generous help in my understanding of properties of additive semigroups and the theory of partitions, and to Sanguthevar Rajasekaran for pointing out to the author some facts of number theory. Finally, the

author has greatly benefited from valuable discussions with Leonard Pitt and Manfred Warmuth, particularly in regard to the subject of 'prediction preserving reducibility', and the learnability results on modules.

References

[1] Naoki Abe. *Money Changing Problem is NP-Complete.* Technical Report MS-CIS-87-45, University of Pennsylvania, June 1987.

[2] George Andrews. *The Theory of Partitions.* Addison-Wesley, 1976.

[3] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Classifying learnable geometric concepts with the vapnik-chervonenkis dimension. In *Proc. 18th ACM Symp. on Theory of Computation*, pages 243 – 282, 1986.

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377 – 380, 1987.

[5] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233 – 235, 1979.

[6] A. Ehrenfeucht, D. Haussler, M. Kearns, and Leslie G. Valiant. A general lower bound on the number of examples needed for learning. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 139 – 154, 1988. Also to appear in Information and Computation.

[7] Michael A. Garey and David S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[8] D. Haussler, N. Littlestone, and M. K. Warmuth. Predicting $\{0,1\}$ functions on randomly drawn points. In *Proceedings of 1988 IEEE Symposium on the Foundations of Computer Science*, 1988. Also in Proceedings of the 1988 Workshop on Computational Learning Theory.

[9] David Helmbold, Robert Sloan, and Manfred Warmuth. Bootstrapping one-sided learning. 1989. To appear.

[10] Michael Kearns and Leslie G. Valiant. *Learning Boolean Formulae or Finite Automata is as Hard as Factoring.* Technical Report TR-14-88, Aiken Computation Laboratory, Harvard University, 1988.

[11] Albert Nijenhuis and Herbert S. Wilf. Representation of integers by linear forms in nonnegative integers. *Journal of Number Theory*, 4:98–106, 1972.

[12] R. J. Parikh. Language generating devices. *M.I.T. Res. Lab. Electron. Quart. Prog. Rep.*, 60:199–212, 1961.

[13] Leonard Pitt. personal communication.

[14] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[15] Leonard Pitt and Manfred Warmuth. *The minimum consistent DFA problem cannot be approximated within any polynomial.* Technical Report UIUCDCS-R-89-1499, University of Illinois at Urbana-Champaign, February 1989.

[16] Leonard Pitt and Manfred Warmuth. Prediction preserving reducibility. 1989. To appear in Journal of Computer and System Sciences.

[17] Leslie G. Valiant. Learning disjunctions of conjunctions. In *The 9th IJCAI*, 1985.

[18] Leslie G. Valiant. A theory of the learnable. *Communications of A.C.M.*, 27:1134–1142, 1984.