SIAM J. COMPUT.
Vol. 21, No. 2, pp. 240–266, April 1992

004

LEARNING INTEGER LATTICES*

DAVID HELMBOLD†, ROBERT SLOAN‡, AND MANFRED K. WARMUTH†

Abstract. The problem of learning an integer lattice of $\mathbf{Z}^k$ in an on-line fashion is considered. That is, the learning algorithm is given a sequence of k-tuples of integers and predicts for each tuple in the sequence whether it lies in a hidden target lattice of $\mathbf{Z}^k$. The goal of the algorithm is to minimize the number of prediction mistakes. An efficient learning algorithm with an absolute mistake bound of $k + \lfloor k \log(n\sqrt{k}) \rfloor$ is given, where n is the maximum component of any tuple seen. It is shown that this bound is approximately a $\log \log n$ factor larger than the lower bound on the worst case number of mistakes given by the VC dimension of lattices that are restricted to $\{-n, \cdots, 0, \cdots, n\}^k$.

This algorithm is used to learn rational lattices, cosets of lattices, an on-line word problem for abelian groups, and a subclass of the commutative regular languages. Furthermore, by adapting the results of [D. Helmbold, R. Sloan, and M. K. Warmuth, *Machine Learning*, 5(1990), pp. 165–196], one can efficiently learn nested differences of each of the above classes (e.g., concepts of the form $c_1 - (c_2 - (c_3 - (c_4 - c_5)))$, where each c_i is the coset of a lattice).

Key words. lattices, concept learning, mistake bounds

AMS(MOS) subject classifications. 68T05, 68Q25, 06B99

1. Introduction.

Integer lattices are one of the most basic combinatorial structures. An integer lattice is a nonempty set of k-tuples from $\mathbf{Z}^k$ that is closed under addition and subtraction. Let $\mathcal{L}^k$ be the concept class consisting of all integer lattices in $\mathbf{Z}^k$. In this paper we present an algorithm for learning $\mathcal{L}^k$, and prove that its performance is within a log log factor of optimal in the on-line model of learning, using the worst case number of mistakes as the performance criterion. Note that any learning algorithm with a good on-line mistake bound can be used as a subroutine to construct a PAC learning algorithm [17], [4], but some PAC learning algorithms have very poor mistake bounds.

Although the learning algorithm is of interest itself, this paper's major technical contributions are analyzing the learning performance of that algorithm and computing the VC dimension for the class of integer lattices (thus giving a lower bound on the worst case number of mistakes made by any learning algorithm). In particular, we prove that our learning algorithm has an absolute mistake bound of $k(1+\log(n\sqrt{k}))$,[1] where n is an upper bound on the absolute value of any component of any instance seen, and that no learning algorithm can have a mistake bound of less than $(1-\epsilon)k \ln n / \ln \ln n$ for any $\epsilon > 0$. Thus we achieve nearly optimal learning performance in a very strict model.[2]

* Received by the editors July 30, 1990; accepted for publication January 15, 1991.

† Department of Computer and Information Sciences, University of California, Santa Cruz, California 95060. The research of the first author was supported in part by a Regents' Junior Faculty fellowship and by Office of Naval Research grant N00014-85-K-0554. The research of the third author was supported by Office of Naval Research grant N00014-86-K-0454. Part of this work was done while the third author was visiting Aiken Computation Laboratory, Harvard University, and was supported by Office of Naval Research grant N00014-85-K-0554.

‡ Electrical Engineering and Computer Science Department, Box 4348, University of Illinois, Chicago, Illinois 60680. Much of this research was done while this author was visiting the Department of Computer and Information Sciences, University of California, Santa Cruz, and was supported by Office of Naval Research grant N00014-86-K-0454. This author's research was also supported by United States Army Research Office grant DAAL 03-86-K-0171 and by a National Science Foundation Graduate Fellowship.

[1] Throughout, log represents the base two logarithm, and ln represents the natural logarithm.

[2] Abe [1] considers learning the harder class of semilinear sets. Using different parameters, he presents a learning algorithm for semilinear sets when $k = 2$.

The algorithm we present keeps a basis for the "smallest" lattice containing all positive examples seen so far and predicts on new instances according to whether they are in this lattice or not. Although any reasonable basis can be used, our algorithm keeps a special basis which facilitates prediction and updates while storing only a small number of derived examples.

Our algorithm is similar to the algorithms of Kannan and Bachem [15] and Chou and Collins [25] for converting an integer basis into a special integer basis called a Hermite normal form (HNF). However, in our application we are not given all the vectors (positive instances) in advance, so we must convert the HNF algorithm into an on-line algorithm. Significant adaptations of the HNF algorithm and its analysis were required for our application, since we want to keep a HNF-like basis while efficiently processing new examples.

We also present several applications of the learning algorithm for $\mathcal{L}^k$. For example, it can be used to learn rational lattices, cosets of lattices, and an on-line word problem for abelian groups. Furthermore, we can use it to learn the subclass of commutative regular languages accepted by DFAs whose single final state reaches the start state. This subclass includes the class of "counter languages." The last result is surprising, because the counter languages are precisely the regular languages used to prove that the minimum consistent DFA problem for regular languages cannot be approximated within any polynomial [21] (see §6.5 for details). Finally, adapting the results from a companion paper [14], our algorithm can be applied to efficiently learn nested differences of all the above learnable classes.

2. Methods and outline. The setting we will be concerned with is that of on-line learning. Formally, *concepts* are subsets of some *instance space* X from which instances are drawn and a *concept class* is a subset of 2^X, the power set over X. The instances are labeled *consistently* with a fixed *target concept* c which is in the *concept class* C to be learned; i.e., an instance is labeled "+" if it lies in the target concept and "−" otherwise. Labeled instances are called *examples*. An *on-line learning algorithm* interactively participates in a series of *trials*. On each trial, the algorithm gets an instance and predicts what its label is. After predicting, the on-line algorithm is informed of the instance's true label. A *mistake* is a trial where the on-line algorithm makes an incorrect prediction. Between trials, the algorithm's *hypothesis* is the subset of the instance space where the algorithm predicts "+."

2.1. Example: Learning vector subspaces. Imagine for the time being that our instance space is an arbitrary vector space $\mathcal{S}$, and that our concept class is the set of all vector subspaces of $\mathcal{S}$. A natural on-line learning algorithm for this problem, shown to us by Haussler and inspired by a similar algorithm of Shvaytser [23] is described in Fig. 1.

2.2. Algorithm V is a closure algorithm. The class of all subspaces of a vector space is intersection-closed, and Algorithm V is in fact a special case of an algorithm for learning intersection-closed concept classes called the "closure algorithm" [20], [6], [14].

DEFINITION. A concept class is *intersection-closed* if for any finite set contained in some concept, the intersection of all concepts containing the finite set is also a concept in the class.

DEFINITION. Fix an intersection-closed concept class $\mathcal{C}$ on some instance space. Let S be a set of positive examples of some concept from $\mathcal{C}$. The *closure* of S with respect to the concept class $\mathcal{C}$ (written CLOSURE(S) when $\mathcal{C}$ is understood) is the

Algorithm V

Predict that the zero vector is positive and all other vectors are negative until a mistake is made.
Whenever a mistake is made predicting the label of some instance, add that instance to a hypothesized basis set for the target subspace.
In general, predict that any instance in the span of the hypothesized basis is positive, and all other instances are negative.

FIG. 1. *An algorithm for learning subspaces of a vector space.*

intersection of all concepts in $\mathcal{C}$ containing the instances of S.

Thus the closure of a set of instances is the smallest concept containing all those instances. The *closure algorithm* is a generic on-line learning algorithm whose hypothesis is CLOSURE(POS) where POS is the set of positive examples seen so far. If the instance space is a vector space, and the concept class is the set of all subspaces, then the closure of a set of positive examples (vectors) is their span. Thus Algorithm V is a closure algorithm.

2.3. How good is closure algorithm V for learning subspaces? This paper uses the mistake-based performance criteria of Littlestone [16] to measure the performance of on-line learning algorithms. For any learning algorithm Q and target concept c define $M_Q(c)$ to be the maximum number of mistakes that Q makes on any possible sequence of instances labeled according to c. For any nonempty concept class C, define $M_Q(C) = \max_{c \in C} M_Q(c)$. Any bound on $M_Q(C)$ is called an (*absolute*) *mistake bound* for algorithm Q applied to class C. The *optimal mistake bound* for concept class C, $opt(C)$, is the minimum over all learning algorithms Q of $M_Q(C)$.

One lower bound on the mistake bound of any learning algorithm for a concept class is given by a very useful combinatorial parameter, the Vapnik–Chervonenkis (VC) dimension [24].

DEFINITION. A set of instances S is *shattered* (by the concept class C) if for each subset $S' \subseteq S$, there is a concept $c \in C$ that contains all of S', but none of the instances in $S - S'$. The *Vapnik–Chervonenkis dimension* of concept class C, denoted by $\mathrm{VCdim}(C)$, is the cardinality of the largest set shattered by the concept class.

Littlestone [16] has given the following relationship between the $\mathrm{VCdim}(C)$ and the optimal mistake bound.

THEOREM 2.1. *For every concept class C,* $\mathrm{VCdim}(C) \leq opt(C)$.

It is fairly easy to see that the closure algorithm V, has an absolute mistake bound of the vector space dimension of $\mathcal{S}$ when learning subspaces of vector space $\mathcal{S}$. Each time a mistake is made, a new vector is added to the basis set Algorithm V maintains, and this can happen at most as many times as the vector space dimension of $\mathcal{S}$. The VC dimension of this concept class is also the vector space dimension of $\mathcal{S}$, so in this case the closure algorithm is optimal.

2.4. The closure algorithm is not always good. Even if a concept class is not intersection-closed, it can always be embedded in one that is. Thus the closure algorithm applies to any concept class. However, there are a number of potential problems:

1. There might not be an efficient algorithm for computing the closure of a set of instances.

2. Given a representation of the closure, it might be difficult to predict whether a new instance is in the closure.
3. The closure algorithm might not have a good mistake bound.

For example, regular sets are intersection-closed. The closure of a finite set of words equals the set itself and thus the closure algorithm makes a mistake on each new positive instance. So here the first two problems do not arise, but the third problem does. If we restrict the concept class to regular sets representable by DFAs with at most s states, then the class is no longer intersection-closed.

Theorem 2.1 implies that any algorithm makes at least as many mistakes as the VC dimension. However, even when the concept class is intersection-closed and has a small VC dimension, the closure algorithm can still make a large number of mistakes. For example, if the instance space is $0, \cdots, n$ and the concepts are initial segments (i.e., the intervals $[0, i]$ for $1 \leq i \leq n$), then the mistake bound of the closure algorithm is n (consider increasing sequences of positive examples) even though the concept class has VC dimension 1.

2.5. Applying the closure algorithm to integer lattices. As in the case of vector spaces, the concept class $\mathcal{L}^k$ of integer lattices of $\mathbf{Z}^k$ is intersection-closed. For any set $S \subseteq \mathbf{Z}^k$, the closure of S is the set of all k-tuples produced by summing integer multiples of elements in S. Thus the generic closure algorithm can be used to learn $\mathcal{L}^k$. However, lattices are significantly more complicated than vector spaces. Any implementation of the closure algorithm must take into account the "holes" in the lattice. For example, consider the lattice generated by the basis $(2, 2)$ and $(0, 2)$. Although the point $(1, 2)$ can be written as $\frac{1}{2}(2, 2) + \frac{1}{2}(0, 2)$, it is not in the lattice, as it is not an *integer* linear combination of the basis vectors. Determining if a point is in a lattice involves solving a set of linear Diophantine equations rather than inverting a matrix.

We give a particular implementation of the closure algorithm, called Algorithm A, which overcomes the three potential problems stated in the previous section.

1. Algorithm A (presented in the Appendix) efficiently computes closures with respect to $\mathcal{L}^k$.
2. Algorithm A represents closures so that prediction can be done efficiently.
3. We prove a good mistake bound for the closure algorithm when applied to lattices.

We show in §3 below that $\text{VCdim}(\mathcal{L}^k)$ is infinite. Thus we know by Theorem 2.1 that the mistake bound of any algorithm must be infinite. To overcome this difficulty, we restrict our attention to $\mathcal{L}^k(n)$, which we define to be the class $\mathcal{L}^k$ where instances are restricted to $\{-n, \cdots, 0, \cdots, n\}^k$. The concept class $\mathcal{L}^k(n)$ is also intersection-closed, as is the restriction of any intersection-closed class to some subset of its domain. Theorem 3.3 below shows that $\text{VCdim}(\mathcal{L}^k(n))$ is roughly $k \ln n / \ln \ln n$.

In this paper we prove that the absolute mistake bound of the closure algorithm when learning $\mathcal{L}^k(n)$, $M_{\text{CLOSURE}}(\mathcal{L}^k(n))$, is at most $k + \lfloor k \log(n\sqrt{k}) \rfloor$. This bound is only a factor of roughly $\log \log n$ larger than the lower bound on the worst case number of mistakes given by the VC dimension of $\mathcal{L}^k(n)$.

Remark. We will not actually limit the size of the input to the closure algorithm; we will simply be describing its performance as a function of the size of its inputs.

2.6. Our algorithm and its advantages. Algorithm A (fully described in the Appendix) for learning $\mathcal{L}^k(n)$ is a particular implementation of the closure algorithm since its hypothesis is always the closure of the positive examples seen so far. Thus Algorithm A has the same mistake bound as the generic closure algorithm. Our

algorithm, however, has some space and computational advantages over the obvious implementations of the closure algorithm.

One obvious implementation of the closure algorithm saves all previous mistakes. After each mistake it recomputes a lower triangular basis for the smallest lattice containing the positive examples using the HNF algorithms of [15], [25]. Prediction is done by back substitution in this lower triangular matrix. Note that the number of mistakes can be at least as large as the VC dimension. As we shall see in §3, the VC dimension is bounded below by approximately $k \ln n / \ln \ln n$, thus this obvious implementation of the closure algorithm requires storing about $k \ln n / \ln \ln n$ positive examples.

In contrast, Algorithm A stores only the lower triangular basis (k derived positive examples). Rather than recomputing the basis from scratch after each mistake, it is updated on-line. The analysis of Algorithm A is nontrivial and is included in the Appendix. One of the more difficult parts is bounding the number of bits required to represent a derived example. The other contribution of this paper is the application of Algorithm A to the problems described in §6.

2.7. Outline of the paper. In the following section we compute the VC dimension of $\mathcal{L}^1(n)$, lattices in one dimension, and bound the VC dimension of $\mathcal{L}^k(n)$. In §4 we compute lower bounds on $opt(\mathcal{L}^k(n))$, the minimum mistake bound any algorithm can achieve when learning $\mathcal{L}^k(n)$. The closure algorithm's mistake bound is then analyzed in §5. At the end of the section, we show that a modified version of the halving algorithm [5], [19], [4] has a nearly optimal mistake bound when learning $\mathcal{L}^1(n)$. Section 6 shows how Algorithm A can be extended to learn rational lattices, cosets of lattices, and a word problem for abelian groups. We conclude §6 by showing how Algorithm A can be applied to learning a subclass of commutative regular languages. In §7 we discuss how Algorithm A can be used in conjunction with a master algorithm for learning nested differences. Our master algorithm is a modification of a similar algorithm presented in a companion paper [14]. It leads to efficient learning algorithms for nested differences of any of the concept classes that we learn using Algorithm A. A short summary of our results is given in the concluding section. In the Appendix we formally state Algorithm A and prove bounds on its resource requirements.

3. VC dimension of integer lattices. This section contains bounds on the VC dimension of the concept class $\mathcal{L}^k(n)$. The VC dimension provides a lower bound on both the number of examples stored by the standard implementation of the closure algorithm [14], and on the mistake bound of any learning algorithm.

We begin by exactly calculating the VC dimension of $\mathcal{L}^1(n)$. Note that each concept in $\mathcal{L}^1(n)$ can be represented by an integer between 0 and n. The concept represented by j contains the subset of $-n, \cdots, n$ whose members are multiples of j.

THEOREM 3.1. *For all $n \geq 1$,*

$$\mathrm{VCdim}(\mathcal{L}^1(n)) = \max\{r \mid 2 \cdot 3 \cdot 5 \cdots p_r \leq 2n\},$$

where p_i is the ith prime.

We start with a definition we will need in our proof.

DEFINITION. Let S be any shattered set; let $T \subseteq S$. We call a concept c such that $T = c \cap S$ a *witness* for T.

Proof. Let r be maximum such that $P = \prod_{i=1}^{r} p_i \leq 2n$ and $d = \mathrm{VCdim}(\mathcal{L}^1(n))$. Now we show $d \geq r$ by exhibiting set S with $|S| = r$ that is shattered: S is all

products of all but one of the first r primes. In symbols, $S = \{P/p_i \mid 1 \leq i \leq r\}$. Since $P/p \leq n$ for all primes p, every element of S is in the instance space. For every $x = P/p_i$ in S, define $\hat{x} = p_i$. It is easy to see that S is shattered: The witness for any nonempty $T \subseteq S$ is $P/\prod_{x \in T} \hat{x} = \prod_{x \notin T} \hat{x}$. The witness for S is 1, and the witness for the empty set is 0.

Hence $d \geq r$. Now we show that $d \leq r$.

Let $S = \{x_1, x_2, \cdots, x_d\}$ be a largest shattered set. First we argue that we may assume that there is no $s > 1$ that divides all elements of S. If there is such an s, we can work instead with $S' = \{x_1/s, x_2/s, \cdots, x_d/s\}$ and divide each witness by s.

Call any subset of $d-1$ elements of S a *minor*. Set S has d minors, S_1 through S_d (where S_i is the minor *not* containing x_i). Let t_i be a witness for S_i.

Now no t_i can be 1, since $t_i \not| x_i$. (Note that $0 \notin S$, because 0 is a positive example of every concept and S is shattered.) Furthermore, $\gcd(t_i, t_j) = 1$ for all $i \neq j$, since $\gcd(t_i, t_j) \mid x$ for every $x \in S$, and we assumed that 1 is the only number with this property. Thus it must be that for each i there is a prime p_i such that $p_i \mid t_i$ but $p_i \not| t_j$ for any $j \neq i$.

Pick any odd $x \in S$. Element x is in $d-1$ minors of S so $d-1$ different t_i's divide x, and x is a multiple of at least $d-1$ different p_i's. Since $x \leq n$ and $2 \not| x$, there are d distinct primes ("2" plus the $d-1$ primes dividing x) whose product is at most $2n$, thus $d \leq r$. □

In order to obtain numerical bounds on the VC dimension, we recall some facts from number theory (see, e.g., Hardy and Wright [13, pp. 262–263]):

1. Let $f(n)$ be the maximum number of consecutive primes such that $\prod_{i=1}^{f(n)} p_i \leq n$. Then for every $\epsilon > 0$, for all sufficiently large n we have

$$f(n) > \frac{(1-\epsilon)\ln n}{\ln \ln n}. \tag{1}$$

2. For all $\epsilon > 0$, for all sufficiently large m, we have

$$\log(\tau(m)) < \frac{(1+\epsilon)\ln m}{\ln \ln m}, \tag{2}$$

where $\tau(m)$ is the number of positive divisors of m.

COROLLARY 3.2. *For all ϵ, for sufficiently large n,*

$$\frac{(1-\epsilon)\ln n}{\ln \ln n} < \mathrm{VCdim}(\mathcal{L}^1(n)) < \frac{(1+\epsilon)\ln n}{\ln \ln n}.$$

Proof. The left inequality follows directly from Theorem 3.1 together with (1), once we note that $\ln n / \ln \ln n < \ln 2n / \ln \ln 2n$.

For the right inequality, we need to bound $f(2n)$, where f is the function specified in (1). Let m be a particular integer of the form $m = 2 \cdot 3 \cdot 5 \cdots p_r$. Note that for such an m we have $\log(\tau(m)) = f(m)$, and thus $\max_{m \leq n} \log(\tau(m)) \geq f(n)$. Thus we have an upper bound on $f(n)$ in terms of the log of the number of divisors of any $m \leq n$, and can apply (2) to get the desired result. □

Remark. The preceding should make it clear that $\mathrm{VCdim}(\mathcal{L}^1(n))$ can be made arbitrarily large by choosing a suitable value for n, and thus that $\mathrm{VCdim}(\mathcal{L}^1)$ is infinite.

We now get a lower bound on the VC dimension of the more general concept class $\mathcal{L}^k(n)$.

THEOREM 3.3.

$$k + \left\lfloor k \log(n\sqrt{k}) \right\rfloor \geq \mathrm{VCdim}(\mathcal{L}^k(n)) \geq k\mathrm{VCdim}(\mathcal{L}^1(n)).$$

Proof. The first inequality is Corollary 5.3, proven later. The proof of the second inequality is essentially a particular case of a bound of Dudley's on the VC dimension of cross products of those concept classes, among them $\mathcal{L}^1(n)$, that have a certain property he calls "being bordered" [9, Thm. 9.2.14].

Let S be the exhibited set of numbers shattered in the proof of Theorem 3.1. Let U be the set of size $k\,|S|$ consisting of all k-tuples of integers containing $k-1$ zeros and one value from S. In this case witnesses are sets of k-tuples. To shatter any $T \subseteq U$, first write $T = T_1 \cup T_2 \cup \cdots \cup T_k$, where all elements of T_i have nonzero values only in position i. For each T_i, let t_i be the number that is the witness for the set of all nonzero components of all elements of T_i (viewed as a concept from $\mathcal{L}^1(n)$ as in the proof of Theorem 3.1). The witness for T is then the set of all integer combinations of the k tuples that have t_i in the ith position and 0's elsewhere. □

The lower bound in Theorem 3.3 is not tight. For example, $\mathrm{VCdim}(\mathcal{L}^1(2)) = 1$, but $\mathrm{VCdim}(\mathcal{L}^2(2)) = 3$ since $(1,2)$, $(2,1)$, and $(2,2)$ are shattered. Determining tight bounds on $\mathrm{VCdim}(\mathcal{L}^k(n))$ remains an open problem.

4. Lower bounds on learning $\mathcal{L}^k(n)$. We know from Theorem 2.1 that $opt(\mathcal{L}^k(n)) \geq \mathrm{VCdim}(\mathcal{L}^k(n))$, so from Theorem 3.3 together with Corollary 3.2, we get a first lower bound for $opt(\mathcal{L}^k(n))$.

COROLLARY 4.1. *For all $\epsilon > 0$, for sufficiently large n,*

$$opt(\mathcal{L}^k(n)) \geq k\,(1-\epsilon)\,\frac{\ln n}{\ln \ln n}.$$

For sufficiently large n we can get a slightly better lower bound on $opt(\mathcal{L}^1(n))$ than $\mathrm{VCdim}(\mathcal{L}^1(n))$ using an adversary argument.[3]

THEOREM 4.2.

$$opt(\mathcal{L}^1(n)) \geq \max_{m \leq n} \sum_{i=1}^{s} \lfloor \log(e_i + 1) \rfloor$$

where $m = \prod_{i=1}^{s} p_i^{e_i}$.

Proof. Let $m = \prod p_i^{e_i}$ be a number less than or equal to n with a maximal number of divisors. Our adversary begins by first giving m as a positive instance. The adversary then makes the algorithm perform a search similar to binary search for the value of each exponent as follows. The next group of instances begin with $p_1^{\lfloor e_1/2 \rfloor} \prod_{i \geq 2} p_i^{e_i}$, and continues with various exponents for p_1 (times $\prod_{i \geq 2} p_i^{e_i}$). The algorithm is forced to make $\lfloor \log(e_1 + 1) \rfloor$ mistakes since there are $e_1 + 1$ choices for the exponent of p_1. The following group of instances all have the exponent of p_1 set to its correct value, the exponents of each p_i for $i \geq 3$ set to e_i, and force the algorithm to search for the value of the exponent of p_2. Next comes a group of instances forcing the algorithm to search for the exponent of p_3, and so on. □

[3] To be precise, the adversary argument gives a stronger bound on $opt(\mathcal{L}^k(n))$ for all $n \geq 2^7 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdots 31 \approx 8.9 \times 10^{13}$. After this point, $\max_{m \leq n} \sum_{i=1}^{s} \lfloor \log(e_i + 1) \rfloor$ is strictly greater than $\max\{r \mid 2 \cdot 3 \cdot 5 \cdots p_r \leq 2n\}$.

COROLLARY 4.3.

$$opt(\mathcal{L}^k(n)) \geq k\left(\max_{m\leq n}\sum_{i=1}^{s}\lfloor\log(e_i+1)\rfloor\right)$$

where $m = \prod_{i=1}^{s} p_i^{e_i}$.

Proof. The method used by the adversary in the proof of Theorem 4.2 can be easily extended to the general case. There are k rounds; in the ith round the adversary gives instances with all components but the ith set to 0. The values of the ith component are chosen according to the adversary strategy in the proof of Theorem 4.2. □

5. Mistake bounds. This subsection calculates $M_{\text{CLOSURE}}(\mathcal{L}^k(n))$, the mistake bound of the closure algorithm, and considers how close it is to $opt(\mathcal{L}^k(n))$. We will bound $M_A(\mathcal{L}^k(n)) = M_{\text{CLOSURE}}(\mathcal{L}^k(n))$ by noticing that every time the algorithm makes a mistake, its new hypothesis is a strict superset of its old hypothesis. At the end of the section we analyze a modified halving algorithm for the special case when $k = 1$.

Before stating the main theorem, we recall some facts about lattices. The standard definition of lattices in $\Re^k$ insists that the lattice be generated by k linearly independent basis vectors. Our definition allows less than k basis vectors, thus our lattices need not have full "rank."

DEFINITION. The *rank* of lattice $\Lambda \subseteq \mathbf{Z}^k$ is the minimum of the ranks of all vector subspaces of $\Re^k$ that contain Λ.

Thus any lattice of rank r can be written as $\{\sum_{i=1}^{r} z_i x_i \mid z_i \in \mathbf{Z}\}$ for some r basis vectors $x_i \in \mathbf{Z}^k$.

It is easy to see that any integer lattice in $\mathbf{Z}^k$ with rank $r < k$ can be rotated into $\Re^r$ (i.e., the last $k-r$ coordinates of every point in the rotated lattice are zeros). For every set of r basis vectors that determine the same lattice, the volume of the r-dimensional parallelepiped that they generate is the same. Furthermore, rotating the lattice into $\Re^r$ preserves volume. The volume of the parallelepiped is called the *determinant* of the lattice [7]. We write $\det\Lambda$ to denote the determinant of lattice Λ. If $\Lambda \subseteq \mathbf{Z}^k$, then $\det\Lambda = 0$ only if Λ is O, the null lattice containing only the origin. Otherwise, $\det\Lambda$ is a positive integer.

THEOREM 5.1. *Let* $O = \Lambda_0 \subset \Lambda_1 \subset \Lambda_2 \subset \cdots \subset \Lambda_m$ *be a sequence of distinct lattices of* $\mathbf{Z}^k$ *where each* Λ_i, *for* $1 \leq i \leq m$, *is the closure of* Λ_{i-1} *plus some* $x_i \in (\mathbf{Z}^k \setminus \Lambda_{i-1})$ *and every component of every* x_i *has absolute value at most* n. *Then*

$$m \leq k + \left\lfloor k\log(n\sqrt{k})\right\rfloor.$$

Proof. There are at most k values of i for which Λ_{i+1} can have greater rank than Λ_i.

Consider now the case where Λ_i and Λ_{i+1} have the same rank. Since Λ_i is a sublattice of Λ_{i+1}, we have $t\det\Lambda_{i+1} = \det\Lambda_i$ for some integer $t \geq 2$ [7].[4] Thus every time the rank of the lattice stays the same, we decrease the determinant by at least a factor of 2.

On the other hand, when the rank of Λ_{i+1} is greater than the rank of Λ_i, then the determinant can increase. The first lattice, Λ_1, is simply all integer multiples of some particular $x_1 \in \mathbf{Z}^k$, so its volume is $\|x_1\| \leq n\sqrt{k}$, where $\|x\|$ denotes the

[4] In geometry of numbers, t is called the index of Λ_i in Λ_{i+1}.