Efficient Learning with Virtual Threshold Gates

Wolfgang Maass*
Manfred K. Warmuth†

UCSC-CRL-95-37
July 28, 1995

Baskin Center for
Computer Engineering & Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

ABSTRACT

We reduce learning simple geometric concept classes to learning disjunctions over exponentially many variables. We then apply an on-line algorithm called Winnow whose number of prediction mistakes grows only logarithmically with the number of variables. The hypotheses of Winnow are linear threshold functions with one weight per variable. We find ways to keep the exponentially many weights of Winnow implicitly so that the time for the algorithm to compute a prediction and update its "virtual" weights is polynomial.

Our method can be used to learn d-dimensional axis-parallel boxes when d is variable, and unions of d-dimensional axis-parallel boxes when d is constant. The worst-case number of mistakes of our algorithms for the above classes is optimal to within a constant factor, and our algorithms inherit the noise robustness of Winnow.

We think that other on-line algorithms with multiplicative weight updates whose loss bounds grow logarithmically with the dimension are amenable to our methods.

---

*Address: Institute for Theoretical Computer Science, Technische Universitaet Graz, Klosterwiesgasse 32/2, A-8010 Graz, Austria. E-mail: maass@igi.tu-graz.ac.at.

†Address: Department of Computer Sciences, University of California, Santa Cruz, CA 95064. E-mail: manfred@cis.ucsc.edu. Supported by NSF grant IRI-9123692.

1 Introduction

We introduce a technique for the design of efficient learning algorithms, that yields superior (and in many cases essentially optimal) learning algorithms for a number of frequently studied concept classes in the most common formal model of on-line learning. In this on-line model [Lit88, Lit89a] learning proceeds on a trial by trial basis. In each trial the algorithm receives an instance and is to produce a binary prediction for that instance. After predicting, the algorithm receives the binary label for the instance w.r.t. the unknown target concept. A *mistake* occurs if the prediction and received label disagree. The mistake bound of an algorithm for a concept class is the worst-case number of mistakes that the algorithm can make on any sequence of examples (instance/label pairs) that are labeled consistently with a target concept in the class. The goal in this model is to find efficient algorithms with small mistake bounds for important concept classes.

The idea of this paper is to reduce learning particular concept classes to the case of learning disjunctions or more generally linear threshold functions over exponentially many variables. Then the algorithm Winnow [Lit88] is applied which learns for example k-literal monotone disjunctions over v variables with a mistake bound of $O(k + k \log(v/k))$. This bound is optimal to within a constant factor since the Vapnik-Chervonenkis dimension [VC71, BEHW89] of the class of k-literal monotone disjunctions is $\Omega(k + k \log(v/k))$ [Lit88] and this dimension is always a lower bound for the optimal mistake bound.

The key feature of Winnow is that its mistake bound grows logarithmically with the number of variables v (when the number of relevant variables k is small). In contrast the number of mistakes of the Perceptron algorithm [Ros58] grows linearly in the number of variables when learning the same concept class of k-literal monotone disjunctions [KW95]. Both the Perceptron algorithm and Winnow actually learn the class of arbitrary linear threshold functions and use linear threshold functions as hypotheses. Monotone disjunctions are a very simple subclass of linear threshold functions: if the instances $\underline{x}$ are v-dimensional Boolean vectors, i.e., $\underline{x} \in \{0,1\}^v$, then the k-literal disjunction $x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k}$ corresponds to the linear threshold function $\underline{w} \cdot \underline{x} \geq \theta$, where $\underline{w}$ is a coefficient vector with $w_{i_1} = w_{i_2} = \cdots = w_{i_k} = 1$ and $w_j = 0$ for $j \notin \{i_1, \ldots, i_k\}$, and the threshold θ is 1.

Good on-line learning algorithms should have mistake bounds that grow polynomially with the parameters of the concept class. That means that the logarithmic growth of the mistake bound of Winnow in the number of variables allows us to use exponentially many variables. For example one can learn Boolean formulas in Disjunctive Normal Form (DNF) using Winnow. Let N be the number of variables of the DNF formula to be learned. Then by introducing one *new* input variable for each of the possible 3^N terms in the DNF formula (i.e. $v = 3^N$) and applying Winnow to the expanded v-dimensional instances one gets a mistake bound of $O(k(N - \log k))$ for k-term DNF formulas over N variables. For this bound the tuning of the parameters of Winnow depends on the number of terms k in the target DNF formula. Note that this mistake bound is optimal to within a constant factor since the VC dimension of k-term DNF is $\Omega(k(N - \log k))$ [DK95]. Also the logarithm of the number of k-term DNF formulas is $O(k(N - \log k))$.

Winnow keeps one weight for each of the v variables and the cost for producing a prediction and for updating the weights after a mistake is $O(1)$ per variable in the straightforward implementation. This makes the above algorithm for learning DNF computationally prohibitive. However the second key feature of our method is to speedup the computation time. We have found cases where we do not need to explicitly maintain the v weights but

still can compute the predictions of Winnow based on the exponentially many "virtual" weights in polynomial time. In the case of DNF such a speedup does not seem to be possible. However for certain geometric concept classes large "blocks" of virtual variables have the same weights and the speedup is obtained by using some clever data structures.

The simplest case where a speedup is possible is the concept class BOX_n^d of d-dimensional axis-parallel boxes with discretized boundaries in $X_n = \{1, \ldots, n\}$ which is defined as follows:

$$\left\{ \prod_{i=1}^{d} \{a_i, a_i + 1, \ldots, b_i\} : a_i, b_i \in X_n \text{ for } i = 1, \ldots, d \right\}.$$

Thus a 2-dimensional box is a rectangle with its sides parallel to the x and y-axis such that the coordinates of the boundaries are integers in X_n.

The complement $\overline{C} = X_n^d - C$ of any concept $C \in \mathrm{BOX}_n^d$ can be represented as the union of the $2d$ halfspaces $\{\underline{x} \in X_n^d : (\underline{x})_i < a_i\}$ for $i = 1, \ldots, d$ and $\{\underline{x} \in X_n^d : (\underline{x})_i > b_i\}$ for $i = 1, \ldots, d$ (where $(\underline{x})_i$ denotes the ith coordinate of the vector $\underline{x}$). We design an efficient learning algorithm for learning the complement $\overline{C}_T$ of an arbitrary rectangle $C_T \in BOX_n^d$ by applying Winnow to a set of $v = 2dn$ Boolean "virtual variables" $u_{i,c}^{<}$ and $u_{i,c}^{>}$ for $i = 1, \ldots, d$ and $c = 1, \ldots, n$. These v new variables are the indicator functions of the $2dn$ halfspaces $H_{i,c}^{<} := \{\underline{x} \in X_n^d : (\underline{x})_i < c\}$ and $H_{i,c}^{>} := \{\underline{x} \in X_n^d : (\underline{x})_i > c\}$. By the preceding observation any $\overline{C}$ for $C \in BOX_n^d$ can be represented as the disjunction of $2d$ of these new variables. Hence we can exploit the advantage of Winnow which is able to achieve a good mistake bound when "irrelevant variables abound" [Lit88].

Note that it takes $\log n$ bits to describe one of the discretized boundaries. Thus a reasonable goal is to aim for a computation time of the learning algorithm that is polynomial in d and $\log n$. Hence we cannot afford to actually run Winnow for the previously described set of $v = 2dn$ variables whose number is exponential in the number of bits it takes to encode a boundary. Instead, we treat the v variables as "virtual" variables and never explicitly compute the v-dimensional weight vector. Using our data structures we keep track of various dynamically changing blocks of variables that currently have the same weight. After every incorrect prediction the number of these blocks increases. Hence it is essential for this approach that the worst-case number of mistakes that Winnow makes grows only logarithmically in the number v of virtual variables. This allows us to bound the number of blocks that ever have to be considered by a polynomial in d and $\log n$ and we can learn the concept class BOX_n^d with a mistake bound of $O(d \log n)$. Our algorithm requires $O(d(\log d + \log\log n))$ time for computing a prediction and updating the data structure representing the virtual weights after a mistake occurs.

This algorithm for learning BOX_n^d is optimal in a very strong sense since it has been shown that any algorithm for learning this class must make $\Omega(d \log n)$ mistakes even if the algorithm is given unbounded computational resources [MT92]. Note that if the hypotheses of the on-line algorithm are required to lie in BOX_n^d, then the best known bounds for learning this class are $O(d^2 \log n)$ [CM92] and $\Omega(\frac{d^2}{\log d} \log n)$ mistakes [Aue93].

Our methods are particularly useful when we are trying to learn lower dimensional subboxes that don't span all d dimensions. Such situations naturally arise when the original instances are expanded to a large number of basis functions and the target boxes are defined in terms of few basis functions. Winnow was designed for applications "when irrelevant attributes abound" [Lit88]. We now can learn boxes when "irrelevant dimensions abound".

Using our methods we can also learn a number of more general geometric concept classes such as unions of up to k boxes: $\bigcup_{\leq k} \mathrm{BOX}_n^d$. The mistake bound and computation time of this algorithm is polynomial if either the number of boxes k or the dimension d is fixed. In this introduction we only state the results for unions of boxes when the dimension is fixed. This concept class has received considerable attention recently [CH95, FGMP94, BCH94, BGGM94]. We can learn $\bigcup_{\leq k} \mathrm{BOX}_n^d$ with a mistake bound of $O(kd \log n)$ and $O((kd \log n)^{2d})$ time for computing a prediction and updating the hypothesis after a mistake occurs. The best previous bounds [BGGM94] were $O((kd \log n)^d)$ mistakes and total computation time. Note that algorithms with mistake or time bounds that have the dimension d in the exponent are of limited interest. The previous bounds have d in the exponent for *both* the mistake bound as well as the time bound. For our algorithm this only happens for the time bound. Moreover we show that our mistake bound is optimal in that it cannot improved by more than a constant factor. The remaining problem of improving the time bound so that it is polynomial in k and d is very ambitious since the concept class of k-term DNF formulas is a special case of $\bigcup_{\leq k} \mathrm{BOX}_n^d$, where $n = 2$, and this would solve a major open problem in Computational Learning Theory: Learning k-term DNF over d variables such that the number of mistakes *and* the time for updating and predicting is polynomial in k and d.

Winnow is robust against noise when learning disjunctions. Our algorithms inherit this noise robustness of Winnow. By using balanced trees the algorithm for learning BOX_n^d can be made very efficient. It is interesting to compare our algorithm with a previous algorithm due to Auer which also learns the class BOX_n^d in the presence of noise [Aue93]. The hypotheses of the latter algorithm are required to lie in target class BOX_n^d, whereas the hypotheses of our algorithm usually lie outside of the target class. The additional requirement leads to larger mistake bounds (at least cubic in the dimension d for Auer's algorithm). His algorithm also applies a simple on-line algorithm related to Winnow called the Weighted Majority algorithm [Lit95, LW94] and uses the same set of virtual variables as our application of Winnow for learning BOX_n^d. Furthermore the virtual variables are maintained as blocks as done in this paper.

Other learning models: So far we have used the most common formal model for on-line learning introduced by Nick Littlestone [Lit88] where the algorithm is to predict on unseen examples with the goal of minimizing the number of prediction mistakes. This model can easily be shown to be the same as Angluin's model for on-line learning with equivalence queries [Ang88] where each mistake corresponds to a query that is answered negatively (we refer to [MT92] for a survey of these and related formal models for on-line learning). In Littlestone's model the hypotheses of the learner are usually not required to be in any particular form, whereas in Angluin's model one distinguishes between *proper* equivalence queries (the hypotheses of the query must be in the target class) and *arbitrary* equivalence queries (the hypotheses of the queries can be arbitrary). The hypotheses of the algorithms developed in this paper are efficiently evaluatable but they are always more general than the target class and thus when translated to Angluin's model they correspond to generalized equivalence queries.

There are standard conversion methods [Lit89b] for translating an on-line algorithm with a worst-case mistake bound to a learning algorithm for the PAC-model [Val84]: If the mistake bound is M then the corresponding PAC-algorithm has sample complexity $O((1/\epsilon)(M + \log(1/\delta))$. However when all examples are given to the learning algorithm at once (as in the PAC-model), then there exists an alternative simple method for learning

concept classes of the type that are considered in this paper. In these concept classes each concept may be viewed as a union of concepts from a simpler concept class and hence learning can be reduced to learning disjunctions of concepts from these simpler concept classes. The standard PAC-algorithm for learning disjunctions is a simple greedy covering algorithm [BEHW89, Hau89, KV94]. This algorithm has the advantage that its hypothesis is a disjunction (but not of minimal size). The best sample size bound obtained for learning k-literal monotone disjunctions over v variables in the PAC model [KV94] with the greedy algorithm is $O((1/\epsilon)(k\log(v)\log(1/\epsilon) + 1/\delta)$. Winnow together with the conversion of [Lit89b] leads to the better bound of $O((1/\epsilon)(k + k\log(v/k) + 1/\delta)$. Thus it is feasible that our reductions to Winnow will also lead to better sample complexity bounds in the PAC model when the hypothesis class is allowed to be larger than the concept class to be learned. However the improvements would only eliminate log factors.

Outline of the paper: After some preliminaries (Section 2) we describe the versions of Winnow that we use in our reductions (Section 3). The second version allows noise in the data. So an added bonus of our method is that our algorithms can tolerate noise. In Section 5 we apply our methods to the case of learning lower dimensional boxes. The next section contains our results for the case when the dimension is variable. We give algorithms for single boxes and unions of a constant number of boxes. In Section 6 we learn unions of boxes when the dimension is fixed. In the last section we discuss how to apply our methods to other learning problems.

2 Preliminaries

A learning problem is given by an *instance* domain X and a family of subsets $\mathcal{C}$ of this domain called *concepts*. *Examples* are $\{0,1\}$-labeled instances. A concept $C \in \mathcal{C}$ is a subset of the domain as well as an indicator function: for an instance $x \in X$, $C(x)$ is one if $x \in c$ and zero otherwise. A sequence of examples is labeled *consistently* with a concept C if all labels of the examples agree with the indicator function C.

3 The Winnow algorithms

The results of this paper use two versions of Littlestone's algorithm Winnow. If the number of variables is v, then the algorithms keep a v-dimensional weight vector of positive weights. Furthermore the algorithms have two parameters: a threshold $\Theta \in \mathbf{R}$ and an update factor $\alpha > 1$. For a given instance Boolean $\langle x_1, \ldots, x_v \rangle \in \{0,1\}^v$ the algorithms predict one iff $\sum_{i=1}^{v} x_i w_i \geq \Theta$. Thus the hypotheses of these learning algorithms are linear threshold functions over $\{0,1\}^v$. For all our results we assume that all weights are initially equal to one.

Assume now that the Winnow1 or Winnow2 make a mistake for some instance $\langle x_1, \ldots, x_v \rangle \in \{0,1\}^v$. If the algorithm predicts 0 and the received label is 1 then both Winnow1 and Winnow2 replace all weights w_i for which $x_i = 1$ by the larger weight $\alpha \cdot w_i$. If the prediction is 1 and the label 0 then Winnow1 replaces all weights w_i for which $x_i = 1$ by 0, whereas Winnow2 replaces these weights by w_i/α. Note that Winnow1 can wipe out weights completely whereas Winnow2 decreases weights more gradually. This makes it possible that Winnow2 can handle noise.

Both algorithms learn the concept class of k-literal monotone disjunction over v variables with a small mistake bound. Such a disjunction is a $\{0,1\}-$valued function on the domain $\{0,1\}^v$ given by the formula $x_{i_1} \vee \ldots \vee x_{i_k}$, where the indices i_j lie in $\{1,\ldots,v\}$. Let $\mathcal{C}_{k,v}$ denote the class of all such formulas. Since the indices are not required to be distinct, $\mathcal{C}_{k,v}$ also contains all disjunctions with less than k literals.

We now state some mistake bounds for the Winnow algorithms [Lit88, Lit89a, Lit91]. They were proven for certain tunings of the parameters α and Θ. In the bounds we give here we let Θ depend on the size of the disjunction k. Only slightly worse bounds can be obtained when the tuning is not allowed to depend on k.

An example $\langle \underline{x}, b \rangle \in \{0,1\}^v \times \{0,1\}$ contains z *attribute errors* w.r.t. a target concept C_T if z is the minimum number of attributes/bits of $\underline{x}$ that have to be changed so that $b = C_T(\underline{x}')$ for the resulting vector $\underline{x}'$. The number of attribute errors for a sequence of examples w.r.t. a target concept is simply the total number of such errors for all examples of the sequence.

(1.1) With $\alpha := 2$ and $\Theta := \frac{v}{2k}$ the algorithm Winnow1 makes at most $2k(1 + \log v/k)$ mistakes on any sequence of examples labeled consistently with any target concept from $\mathcal{C}_{k,v}$.

(1.2) With $\alpha := 1.5$ and $\Theta := v/k$ the algorithm Winnow2 makes at most $4z + 8k + 14k \ln(v/k)$ mistakes on any sequence of examples which has at most z attribute errors w.r.t. some target concept from $\mathcal{C}_{k,v}$.

So the bound for Winnow1 is better but Winnow2 can handle noise. A sequence has q *classification errors* w.r.t. a target concept if q labels have to be flipped so that the sequence is consistent with the target. It is easy to see that each classification error can be compensated by up to k attribute errors if the target C_T is a k-literal disjunction. The theorems of this paper mostly deal with attribute errors. Analogous theorems for classification errors or a mixture of both can be obtained easily using the above observation. Note that the tunings for Winnow2 are independent of the amount of noise in the sequence of examples. So noise simply causes more mistakes but the algorithm does not change.

Note that if the number of attributes that are *relevant* for a target concept from $\mathcal{C}_{k,v}$ (there are at most k of them) is small then the mistake bounds for both versions of Winnow grow only logarithmically in the total number of attributes v. This is essential for the results of this paper where we will reduce the given learning problems to applications of Winnow with exponentially many attributes. The other property of the algorithms that we exploit in this approach is that they change their weights in a very *uniform* manner: all w_i with $x_i = 1$ are multiplied with the *same* factor which is either α, $1/\alpha$ or 0.

Note that the threshold Θ in the above tunings for Winnow1 and Winnow2 depend on k, the maximum size of the target disjunction. There are alternate tunings in which no information is used of the size of the target disjunction. For example Θ can be set to the number of variables v and if α is adjusted appropriately then the main change in the above bounds is that the $\log(v/k)$ terms are replaced by $\log v$ terms and the constants before the summands change [Lit88].

One can also take the route of using more information for tuning the algorithms and let the tunings of the parameters α and Θ depend on k as well as an upper bound Z of the number of attribute errors of the target disjunction [AW95]. In this case it is possible to obtain mistake bounds of the type $2z + (2\sqrt{2} + o(1))\sqrt{Ak \ln(n/k)}$, where $z \leq Z$ is the number of attribute errors of some target disjunction from $C_{k,v}$. The constant of 2 in front of the number of attribute errors z is now optimal and this constant drops to one in the

expected mistake bound of a probabilistic algorithm. For the sake of simplicity we did not state the mistake bounds for the more sophisticated tunings in this paper.

4 Efficient On-line Learning of Simple Geometrical Objects When Dimension is Variable

We first consider learning the concept class BOX_n^d of axis-parallel rectangles over the domain X_n^d. As outlined in the introduction the complement of such boxes can be described as the disjunction of $2dn$ variables that represent halfspaces which are parallel to some axis. It is our goal that the computation time of the learning algorithm is polynomial in d and $\log n$. Hence we cannot afford to actually run Winnow for the set of $v = 2dn$ variables. Instead, we simulate Winnow by keeping track of various dynamically changing blocks of variables that currently have the same weight. After every incorrect prediction the number of these blocks increases by $2d$. Hence it is essential for this approach that the worst-case number of mistakes that Winnow makes grows only logarithmically in the number v of virtual variables. This allows us to bound the number of blocks that ever have to be considered by a polynomial in d and $\log n$ and we can prove the following result.

Theorem 1: *There exists an on-line learning algorithm for BOX_n^d that makes at most $O(d \log n)$ mistakes on any sequence of examples labeled consistently with a target concept from BOX_n^d. This algorithm uses at most $O(d(\log d + \log\log n))$ time for predicting and for updating its data structures after a mistake.*

Before we prove this theorem, we would like to note that this learning algorithm is *optimal* in a rather strong sense [CM92]: using a simple adversary argument one can show that *any* on-line learning algorithm can be forced to make $\Omega(d \log n)$ mistakes on some sequence of examples labeled consistently with a target in BOX_n^d.

For the sake of completeness and since similar methods are used for Theorem 7 we reprove this lower bound here. For a simple start consider the concept class of initial segments on $\{1, \ldots, n\}$. Each initial segment is determined by its right endpoint. We claim that an adversary can force any algorithm to do a binary search for the endpoint which leads to lower bound of $\lfloor \log n \rfloor$ mistakes. We prove this bound as follows. For any set of examples the set of remaining consistent initial segments is characterized by an interval of possible right endpoints. The adversary always chooses its next instance in the middle of the remaining interval and forces a mistake by choosing a label for the instance that disagrees with the algorithm's prediction. Originally the interval is of length n and each example cuts the length of the interval in half. At the end the length of the interval in one and the adversary ends up with an initial segment that is consistent with all the examples.

Similarly for the class of intervals on $\{1, \ldots, n\}$ an adversary can force $2\lfloor \log(n/2) \rfloor$ mistakes, $\lfloor \log(n/2) \rfloor$ for the left and right boundary. For the concept class BOX_n^d the interval argument is repeated for each of the d dimensions. While forcing the two binary searches in one dimension the other dimensions are set to a middle point. This gives an overall lower bound for the class BOX_n^d of $2d\lfloor \log(n/2) \rfloor$ mistakes.

The mistake bound for a concept class is always one less than the maximum number of equivalence queries required for learning the class [Lit88]. By a result from [AL94] it immediately follows that even if membership queries are allowed then the total number of equivalence and membership queries is still $\Omega(d \log n)$. If the hypotheses of the equivalence queries must be boxes in BOX_n^d as well, then this lower bound can be raised to $\Omega(\frac{d^2}{\log d} \log n)$ [Aue93].

Proof of Theorem 1: We now give a detailed description of the technique that was outlined at the beginning of this section. Obviously it suffices to exhibit an efficient on-line learning algorithm for the *complements* $\overline{C} := X_n^d - C$ of arbitrary rectangles $C \in BOX^d$ (in order to turn this into an efficient on-line learning algorithm for BOX_n^d one just has to negate the output-bit for each of its predictions). Assume that the environment has fixed some $\overline{C}_T$ with $C_T \in BOX_n^d$. In order to predict for some arbitrary given $\underline{y} \in \{1,\ldots,n\}^d$ whether $\underline{y} \in \overline{C}_T$ one applies the following variable transformation $\underline{y} \mapsto \underline{u}(\underline{y}) \in \{0,1\}^v$, which reduces this prediction problem to a prediction problem for a "virtual threshold gate" with $v = 2dn$ Boolean input variables.

For each halfspace $H_{i,c}^{<} := \{\underline{x} \in X_n^d : (\underline{x})_i < c\}$ we consider an associated "virtual variable" $u_{i,c}^{<}$, and for each halfspace $H_{i,c}^{>} := \{\underline{x} \in X_n^d : (\underline{x})_i > c\}$ we consider an associated "virtual variable" $u_{i,c}^{>}$ $(i = 1,\ldots,d; c = 1,\ldots,n)$. For any $\underline{y} \in \{1,\ldots,n\}^d$ we set the associated virtual variable $u_{i,c}^{<}$ (resp. $u_{i,c}^{>}$) equal to 1 if $\underline{y} \in H_{i,c}^{<}$ (resp. $H_{i,c}^{>}$), and else equal to 0. This defines the desired variable transformation $\{1,\ldots,n\}^d \ni \underline{y} \mapsto \underline{u}(\underline{y}) \in \{0,1\}^v$ for $v = 2dn$.

One can then apply Winnow1 or Winnow2 to the resulting learning problem over $\{0,1\}^v$. For each $C_T \in BOX_n^d$ and any $\underline{y} \in \{1,\ldots,n\}^d$ the set of virtual variables in $\underline{u}(\underline{y})$ with value 1 forms for each $i \in \{1,\ldots,d\}$ a final segment of the sequence $u_{i,1}^{<},\ldots,u_{i,n}^{<}$ (since $\underline{y} \in H_{i,c}^{<} \Rightarrow \underline{y} \in H_{i,c'}^{<}$ for all $c' > c$), and dually an initial segment of the sequence $u_{i,1}^{>},\ldots,u_{i,u}^{>}$. Assume that $C_T = \prod_{i=1}^{d} \{a_i,\ldots,b_i\}$ with $1 \le a_i \le b_i \le n$ for $i = 1,\ldots,d$. Then $\overline{C}_T = \bigcup_{i=1}^{d} H_{i,a_i}^{<} \cup \bigcup_{i=1}^{d} H_{i,b_i}^{>}$, and one can therefore reduce the problem of on-line learning $\overline{C}_T$ to the problem of on-line learning of the disjunction $\bigvee_{i=1}^{d} u_{i,a_i}^{<} \vee \bigvee_{i=1}^{d} u_{i,b_i}^{>}$ over the $v = 2dn$ variables $u_{i,c}^{<}, u_{i,c}^{>}$, for $i = 1,\ldots,d$ and $c = 1,\ldots,n$. For this reduction one simply takes each prediction for "$\underline{u}(\underline{y}) \in \bigvee_{i=1}^{d} u_{i,a_i}^{<} \vee \bigvee_{i=1}^{d} u_{i,b_i}^{>}$?" of Winnow and uses it as a prediction for "$\underline{y} \in \overline{C}_T$?". This prediction for "$\underline{y} \in \overline{C}_T$?" is incorrect if and only if the prediction for "$\underline{u}(\underline{y}) \in \bigvee_{i=1}^{d} u_{i,a_i}^{<} \vee \bigvee_{i=1}^{d} u_{i,b_i}^{>}$?" is incorrect. Hence the worst-case number of mistakes of the resulting on-line learning algorithm for the complements $\overline{C}_T$ of rectangles $C_T \in BOX_n^d$ is bounded by the worst-case number of mistakes of Winnow for learning a disjunction of $2d$ out of $2dn$ variables.

If one applies Winnow1 with $\alpha = 2$ and $\Theta = v/4d = n/2$, then one obtains a mistake bound $4d(1 + \log n)$ for learning BOX_n^d.

The computation time of our learning algorithm for this simulation of Winnow1 (respectively Winnow2) can be estimated as follows. After s mistakes each group of variables $u_{i,1}^{<},\ldots,u_{i,n}^{<}$ (resp. $u_{i,1}^{>},\ldots,u_{i,n}^{>}$) consists of up to $s+1$ "blocks" $u_{i,a}^{<}, u_{i,k+1}^{<},\ldots,u_{i,b}^{<}$ (respectively $u_{i,a}^{>}, u_{i,k+1}^{>},\ldots,u_{i,b}^{>}$) of variables that currently all have the same weight. This structure arises from the fact that whenever the weight of any of these variables is changed, then all weights of a final or an initial segment of this group of n variables are changed in the same way (i.e. multiplied with the same factor). Of course it suffices to store for each of the $2d$ groups of n virtual variables just the *endpoints* of these up to $s+1$ blocks, together with the current weight of the associated virtual variables.

By the preceding analysis the total number of mistakes s is $O(d \log n)$. Hence without use of a more sophisticated data structure at most $O(d^2 \cdot \log n)$ computation steps (on a RAM) are needed to decide for any given $\underline{y} \in \{1, \ldots, n\}^d$ whether "$\underline{y} \in H$?" for the current hypothesis H, or to update the hypothesis after a mistake. The improved time bound of $O(d(\log d + \log \log n))$ which uses balanced trees is given in the appendix. ∎

By using Winnow2 instead of Winnow1 it is easy to generalize the above theorem to the noisy case. (See [BGGM94] for earlier results on learning similar geometric objects in the presence of noise.) For this purpose the notion of attribute error is generalized in the straightforward way: An example $\langle \underline{x}, b \rangle \in X_n^d \times \{0,1\}$ contains z *attribute errors* w.r.t. a target box C_T in BOX_n^d if z is the minimum number of conponents of $\underline{x}$ that have to be changed so that $b = C_T(\underline{x}')$ for the resulting vector $\underline{x}'$. As before, the number of attribute errors for a sequence of examples w.r.t. a target concept is simply the total number of such errors for all examples of the sequence.

Theorem 2: *There exists an on-line learning algorithm that makes at most $O(z + d \log n)$ mistakes on any sequence that has at most z attribute errors w.r.t. a target concept in BOX_n^d. This algorithm requires $O(d \log(z + d \log n))$ time for predicting and for updating its data structures after a mistake occurs.*

Proof: We proceed exactly as in the proof of Theorem 1, except that we apply Winnow2 instead of Winnow1 to the virtual variables. As indicated in Section 3, Winnow2 tolerates attribute errors in the examples. Hence Theorem 2 follows with the help of the following observation immediately from (1.2) and the proof of Theorem 1. Each single attribute error in an example $\langle \underline{x}, b \rangle$ that occurs in a learning process for some target concept $C_T \in BOX_n^d$ gives rise to at most *one error* in a *relevant* attribute for the transformed learning process, where one learns a disjunction of $2d$ of the $v = 2dn$ virtual variables. Note however that it may give rise to a rather *large* number of errors in *irrelevant* attributes of the transformed sequence of examples. The time bound for predicting and updating is again $O(d \log r)$, where r is the number of mistakes done so far. The argument is given in the appendix. ∎

The above mistake bound grows linearly in the number of attribute errors. The bound immediately leads to a similar theorem for classification errors, since q classification errors correspond to at most $k\,q$ attribute errors when the concept class is k-literal monotone disjunctions. It is well-known that for classification errors there exists for Winnow a trade-off between noise-tolerance and computation time: the factor k before q can be decreased at the expense of a larger computation time. Since this trade-off requires a transformation of the input variables for Winnow, it is not a-priori clear that a similar result (with a not too drastic increase in the computation time) can also be shown for the learning algorithm used for Theorem 2. However the following result shows that our new learning algorithm for BOX_n^d does in fact inherit this attractive feature of Winnow.

Theorem 3: *Let $R \in \mathbf{N}$ be some arbitrary parameter. Then there exists an on-line learning algorithm for BOX_n^d that makes at most $O(d/R + d \log n + q\ d/R)$ mistakes on any sequence of examples that has $\leq q$ classification errors w.r.t. some target in BOX_n^d. This learning algorithm requires $O(R(d/R + d \log n + q\ \ d/R)^R)$ time for predicting and updating its hypothesis after a mistake.*

Proof: Consider the $2d$ groups of virtual variables that were discussed in the proof of Theorem 1. We partition these $2d$ groups into $g := \lceil \frac{2d}{R} \rceil$ classes $B_1, \ldots, B_g$ that each consist of R or less groups of virtual variables. For the sake of simplicity we assume in the following that each of these classes consists of exactly R groups of virtual variables. We then create new "virtual variables" of the types $B_1, \ldots, B_g$. For each $B_j (j \in \{1, \ldots, g\})$

the variables of type B_j represent arbitrary disjunctions of R variables with one arbitrary variable chosen from each of the R groups of virtual variables that belong to class B_j. Hence there are n^R variables of type B_j.

Thus we have created altogether $g\,n^R$ new virtual variables, and each complement of a target concept $C_T \in BOX_n^d$ can be represented as a disjunction of g of these new variables. We then apply Winnow2 with $\Theta := n^R$ and $\alpha := 3/2$ in order to learn arbitrary complements of concepts from BOX_n^d with regard to this new representation. Each classification error is compensated by g attribute errors. Thus according to (1.2) we have that for any sequence of examples which has at most q classification errors w.r.t. a target in BOX_n^d the sketched algorithm makes at most $8g + 14gR\ln n + 4gq$ mistakes.

In order to compute each prediction of Winnow2 in an efficient manner, one exploits that for each of the g types B_j of variables, the variables can be identified with points in the R-dimensional space $\{1,\ldots,n\}^R$. Further whenever a mistake occurs for some example $\langle \underline{y}, b\rangle$, then the set of variables of type B_j whose weight is increased (respectively decreased), forms a union of R orthogonal halfspaces. Hence after r mistakes the n^R variables of type B_j (viewed as points in $\{1,\ldots,n\}^R$) have been partitioned by these unions of halfspaces into up to $(r+1)^R$ axis-parallel "rectangles" of variables so that all variables in the same "rectangle" have the same current weight.

It is convenient to keep for each type B_j of virtual variables the records for these rectangles in lexicographical order with regard to their "leftmost" corner point, and to attach the current common weight of the variables in this rectangle to each of these records. In this way each prediction of Winnow2 and each update of this data structure requires after r mistakes at most $O(R(r+1)^R)$ computation time. ∎

The preceding results can be extended to learning the class of k-fold unions of boxes $\bigcup_{\le k} BOX_n^d$, which can we expressed as follows:

$$\{B_1 \cup \ldots \cup B_{k'} \;:\; k' \le k \text{ and } B_1,\ldots,B_{k'} \in BOX_n^d\}.$$

The following theorem shows that unions of $k = O(1)$ arbitrary boxes from BOX_n^d can be learned by applying Winnow to a virtual threshold gate so that the complexity bounds remain polynomial in d and $\log n$.

Theorem 4: *For any constant k, there is a noise-robust on-line learning algorithm that makes at most $O(d^k \log n + z)$ mistakes on any sequence of examples which has up to z attribute errors w.r.t. some concept in $\bigcup_{\le k} BOX_n^d$. This algorithm uses at most $O(d^k(d^k \log n + z)^k)$ time for predicting and for updating its hypothesis after a mistake.*

Proof: We first consider the $2dn$ axis-parallel halfspaces $H_{i,c}^{<}$ and $H_{i,c}^{>}$ (for $i = 1,\ldots,d$; $c = 1,\ldots,n$) from the proof of Theorem 1. Any concept $C \in \bigcup_{\le k} BOX_n^d$ can obviously be represented as the $2d$-fold union of k intersections of complements of such halfspaces. That is any concept is in the form

$$\left(\bigcap_{i=1}^{d} \overline{H_{i,c_1(i)}^{<}} \cap \bigcap_{i=1}^{d} \overline{H_{i,\tilde{c}_1(i)}^{>}}\right) \cup \ldots \cup \left(\bigcap_{i=1}^{d} \overline{H_{i,c_k(i)}^{<}} \cap \bigcap_{i=1}^{d} \overline{H_{i,\tilde{c}_k(i)}^{>}}\right)$$

with suitable values $c_j(i), \tilde{c}_j(i) \in \{1,\ldots,n\}$. This implies that the *complement* of any $C \in \bigcup_{\le k} BOX_n^d$ can be represented in the form

$$\bigcup_{s=1}^{(2d)^k} \bigcap_{j=1}^{k} H(j,s),$$

where each $H(j,s)$ is a halfspace of the form $H^{<}_{i,c}$ or $H^{>}_{i,c'}$ with certain $i \in \{1,\ldots,d\}$ and $c,c' \in \{1,\ldots,n\}$. Hence we create for each intersection $\bigcap_{j=1}^{k} H(j)$ of k halfspaces of this form a new virtual variable u, which receives the value 1 in the variable transformation $\{1,\ldots,n\}^d \ni \underline{y} \mapsto \underline{u}(\underline{y}) \in \{0,1\}^{(2dn)^k}$ if and only if $\underline{y} \in \bigcap_{j=1}^{k} H(j)$. This yields $(2dn)^k$ virtual variables.

Analogously as in the proof of Theorem 1 one reduces in this way the learning of the complement of an arbitrary target concept $C_T \in \bigcup_{\leq k} BOX^d_n$ to the learning of a disjunction of at most $(2d)^k$ of these $(2dn)^k$ virtual variables. One applies Winnow2 to this new learning problem for a "virtual threshold gate" of size $(2dn)^k$. The desired mistake bound follows directly from (1.2).

For the analysis of the computation time of the resulting learning algorithm for $\bigcup_{\leq k} BOX^d_n$ we observe that the $(2dn)^k$ virtual variables naturally fall into $(2d)^k$ sets of variables of called *types*. We say here that two virtual variables have the same type if they represent two intersections $\bigcap_{j=1}^{k} H(j)$ and $\bigcap_{j=1}^{k} \tilde{H}(j)$ of k halfspaces using the same k-tuple of dimensions and the same k-tuple of orientations of the halfspaces. In symbols this means that for each $j \in \{1,\ldots,k\}$ there exist a common dimension $i \in \{1,\ldots,d\}$, a common orientation $o \in \{<,>\}$, and integers $c,\tilde{c} \in \{1,\ldots,n\}$ such that $H_{(j)} = H^{o}_{i,c}$ and $\tilde{H}_{(j)} = H^{o}_{i,\tilde{c}}$.

After s mistakes the virtual variables of each type are partitioned into up to $(s+1)^k$ k-dimensional "rectangles" of variables that currently have the same weight. The sum of the weights of all variables in any such k-dimensional "rectangle" can be computed by computing in $k = O(1)$ computation steps its "volume", and by multiplying this volume with the common weight of these variables. According to (1.2), Winnow2 makes at most $s = O(d^k + d^k \log n + z)$ mistakes on any sequence of examples which has up to z attribute errors w.r.t. some concept in $\bigcup_{\leq k} \mathrm{BOX}^d_n$. Thus the time for predicting and updating the weights after a mistake is $O((2d)^k(s+1)^k)$ which is $O(d^k(d^k \log n + z)^k)$. ∎

5 Learning Lower-Dimensional Boxes

Winnow was designed for learning disjunctions when the size of the disjunction is small compared to the number of variables. Similarly, one might want to learn lower dimensional boxed (i.e. boxes that depend on only few variables). This is particularly useful when the original instances are expanded to a large number d of basis functions and the dimension of the target boxes are much smaller than d (only a small number of the basis functions are used in the target concept).

This leads to the following definition and theorem. For $u \leq d$ a u-dimensional subbox of BOX^d_n is given by u dimensions $i_j \in \{1,\ldots,d\}$ and two boundaries $a_{i_j}, b_{i_j} \in X_n$ per dimension i_j. The corresponding box is defined as

$$\left\{\underline{x} \in X^d_n \;:\; \forall 1 \leq j \leq u : a_{i_j} \leq x_{i_j} \leq b_{i_j}\right\}$$

Let $\mathrm{BOX}^{u,d}_n$ denote the set of all such boxes.

Theorem 5: *There exists an on-line learning algorithm that makes at most $O(z + u\log(dn/u))$ mistakes on any sequence that has at most z attribute errors w.r.t. a target concept in $BOX^{u,d}_n$. This algorithm requires $O(d\log(z + u\log(dn/u)))$ time for predicting and for updating its data structures after a mistake occurs.*

Proof: As in Theorem 2 use Winnow2 with $2dn$ variables. Complements of Boxes in $\mathrm{BOX}_n^{u,d}$ become disjunctions of size $2u$ over these variables. ∎

One of the most basic applications of our method of reducing learning problems to disjunctions over exponentially many virtual variables leads a learning algorithm for the following simple generalization of k out v literal monotone boolean disjunctions. Here the base variables are non-boolean and lie in the domain X_n. The generalized disjunctions are mappings from X_n^v to $\{0,1\}$ given by the formulas of the form

$$(x_{i_1} \leq a_1) \vee \ldots \vee (x_{i_k} \leq a_k),$$

where the indices i_j lie in $\{1,\ldots,v\}$ and the boundaries a_j lie in X_n. Let $\mathcal{LIN}_{k,v}$ the class of all such formulas.

Theorem 6: *There exists an on-line learning algorithm that makes at most $O(z+u\log(dn/u))$ mistakes on any sequence that has at most z attribute errors w.r.t. a target concept in $\mathcal{LIN}_{k,v}$. This algorithm requires $O(d\log(z+u\log(dn/u)))$ time for predicting and for updating its data structures after a mistake occurs.*

Proof: Note that concepts in $\mathcal{LIN}_{k,v}$ are complements of boxes which have the origin as one corner. So we only need to use one variable per dimension v and boundary in X_n. The concepts become k literal monotone disjunctions over the vn variables. In this case no complementation of the concepts is necessary. Again we simulate Winnow2 with the usual data structures. ∎

6 Efficient On-line Learning of Simple Geometrical Objects When Dimension is Fixed

We show that with the same method as in the preceding section one can also design an on-line learning algorithm for $\bigcup_{\leq k} \mathrm{BOX}_n^d$ whose complexity bounds are polynomial in k and $\log n$ provided that d is a constant. We assume that $n \geq 2k$, so that the bounds are easy to state.

Theorem 7: *For any constant dimension d, there exists an on-line learning algorithm for $\bigcup_{\leq k} BOX_n^d$ that makes at most $O(kd\log n + z)$ mistakes on any sequence of examples for which there is a concept in $\bigcup_{\leq k} BOX_n^d$ with at most z attribute errors. The algorithm uses $O((kd\ln n + z))^{2d})$ time for predicting and for updating its hypothesis after a mistake.*

Also any algorithm regardless of computational resources for learning $\bigcup_{\leq k} BOX_n^d$ makes at least $O(kd\log n)$ mistakes on some sequence of examples consistent with a concept in $\bigcup_{\leq k} BOX_n^d$.

Proof: For every point $\underline{\mathbf{p}} := \langle a_1,\ldots,a_d,\ b_1,\ldots,b_d\rangle \in \{1,\ldots,n\}^{2d}$ we introduce a virtual variable $u_{\underline{\mathbf{p}}}$. For any $\underline{y} \in \{1,\ldots,n\}^d$ we assign to this virtual variable in the transformation

$$\{1,\ldots,n\}^d \ni \underline{y} \mapsto \underline{u}(\underline{y}) \in \{1,\ldots,n\}^{2d}$$

the value $u_{\underline{\mathbf{p}}}(\underline{y}) = 1$ if and only if $\underline{y} \in \prod_{i=1}^{d} \{a_i,\ldots,b_i\}$.

Obviously any target concept $C_T \in \bigcup_{\leq k} BOX_n^d$ can be represented as a disjunction of up to k of the n^{2d} virtual variables. According to (1.2) at most $O(k + k\log(n^{2d}/k) + z)$ mistakes can occur when one applies Winnow2 to the transformed learning problem. This mistake bound is $O(kd\log n + z)$ for $n \geq 2k$.

Furthermore if the resulting learning algorithm for $\bigcup_{\leq k} BOX_n^d$ makes an incorrect prediction for some example $\underline{y} \in \{1,\ldots,n\}^d$, then the set of virtual variables $u_{\underline{\mathbf{p}}}$ with $u_{\underline{\mathbf{p}}}(\underline{y}) = 1$ forms a "rectangle" in $\{1,\ldots,n\}^{2d}$: this set consist of those virtual variables $u_{\underline{\mathbf{p}}}$ that are associated with vectors $\underline{\mathbf{p}} = \langle a_1,\ldots,a_d,b_1,\ldots,b_d\rangle \in \{1,\ldots,n\}^{2d}$ such that $\underline{y} \in \prod_{i=1}^{d}\{a_i,\ldots,b_i\}$, i.e. $a_i \leq (\underline{y})_i \leq b_i$ for $i = 1,\ldots,d$. Hence after r mistakes the virtual variables are partitioned into $\leq (r+1)^{2d}$ rectangles of equal weight over the domain $\{1,\ldots,n\}^{2d}$.

It is easy to predict in time linear in the current number of rectangles. With some simple data structures one can also update the list of rectangles in time linear in the number of rectangles that exist after the update is completed. (Note that the dimension of the rectangles is assumed to be constant.)

The lower bound is proven using an adversary argument that is similar to the one used for BOX_n^d (See comments after the statement of Theorem 1). For the concept class $\bigcup_{\leq k}\mathrm{BOX}_n^d$ the adversary first forces $2k\lfloor\log(n/2k)\rfloor$ mistakes to fix the k intervals of the boxes in the first dimension. This is done by forcing $2k$ binary searches over ranges of size $\lfloor n/2k\rfloor$ for each of the $2k$ boundaries in the first dimension. The first box's interval lies in $\{1,\cdots,2\lfloor n/2k\rfloor\}$, the interval of the second box in $\{2\lfloor n/2k\rfloor+1,\cdots,4\lfloor n/2k\rfloor\}$, and so forth. Since the k rectangles are already disjoint in the first dimension, the searches in the remaining dimensions can start with a range of size $n/2$. In total the adversary can force at least $2k\cdot\lfloor\log\frac{n}{2k}\rfloor + 2(d-1)k\lfloor\log\frac{n}{2}\rfloor$ mistakes, which is $\Omega(kd\log n)$ when $n \geq 2k$. ∎

7 Conclusions

There are a number of algorithms that can learn k-literal monotone disjunctions with roughly the same mistake bound as Winnow: the Balanced algorithm [Lit89a] and the Weighted Majority algorithm [Lit95, LW94]. All of them maintain a linear threshold function and do multiplicative weight updates. It is likely that the results of this paper can also be obtained if we use these other algorithms for the reductions of this paper in place of Winnow. The Weighted Majority algorithm is in some sense the simplest one since its weights are only multiplied by one factor instead of two. We chose Winnow since for the purpose of learning disjunctions it is the most studied of the group.

Winnow is robust against malicious attribute noise and our reductions preserve these properties. Slight modifications of Winnow have shown to give good mistake bounds in relation to the best shifting disjunction [AW95]. By combining these recent results with the findings of this paper one immediately obtains an algorithm with a small mistake bound compared to the best shifting box.

Mistake bounds for Winnow have also been developed for j-of-k threshold functions. Such functions are one when at least j out of a subset k of the v literals are one. Disjunctions are 1-of-k threshold functions. Using these additional capabilities of Winnow we get for example an algorithm for learning the following concept class with a good mistake bound: A concept is defined by k boxes in BOX_n^d and an instance in X_n^d is in the concept if it lies in at least j of the k boxes. Using the reduction of Theorem 7, Winnow2 when suitably tuned makes at most $O(j^2 + jkd\log n + z)$ mistakes on any sequence of examples that has at most z attribute errors w.r.t. one of such concept. The algorithm is again noise robust and its time bound for predicting and updating its hypothesis remains $O(r^{2d})$, where r is current number of mistakes.

In this paper the most basic geometric objects we considered were axis-parallel boxes over the discretized domain $\{1,\ldots,n\}^d$. Instead we could have defined boxes and other geometric objects corresponding to $\bigcup_{\leq k} \mathrm{BOX}_n^d$ in terms of an arbitrary set of directions $\mathcal{D} \in \mathbf{R}^d$ (see e.g. [BCH94]). The basic virtual variables would then correspond to the following halfspaces over the domain $\mathbf{R}^d$:

$$\{\underline{x} \in \mathbf{R}^d : \underline{x} \cdot \underline{a} < \Theta\} \text{ or } \{\underline{x} \in \mathbf{R}^d : \underline{x} \cdot \underline{a} > \Theta\} \text{ where } \Theta \in \{1,\ldots,n\} \text{ and } \underline{a} \in \mathcal{D}\}.$$

It is easy to apply our methods to this case by simply changing the transformation to the virtual threshold gate. The key ingredient is that the concepts class to be learned can be reduced to small disjunctions over an exponentially large set of virtual variables so that Winnow can still be simulated efficiently. It would also be interesting to form virtual variables from past examples. For example when the dimension is fixed and m examples have been seen so far, then subsets of size d (i.e. $\binom{m}{d}$ of them) determine hyperplanes (halfspaces) that corresponding a useful set of virtual variables.

The class of axis-parallel boxes is a simple example of an *intersection-closed* concept class and nested differences of concepts from this class are efficiently learnable in the PAC model [HSW90]. A challenging open problem is to find an *on-line* algorithm for learning nested differences of axis-parallel boxes over the discretized domain $\{1,\ldots,n\}^d$ with at most $O(pd\log n)$ mistakes (where p is the depth of the target concept) and time polynomial in p, d, and $\log n$.

There is a large family of on-line algorithms (besides Winnow and its relatives) with multiplicative weight updates whose loss bounds grow logarithmically with the dimension of the problem [Vov90, HKW94, KW94]. We expect that further applications will be found where these algorithm can be simulated for exponentially many "virtual variables". In parallel work such applications have been found in [HW95, HS95, AKMW95]. A more challenging goal is to apply this family of algorithms to continuously many variables. See Cover [Cov91] for an example problem for which this was done.

Acknowledgements

We would like to thank Sally Goldman for valuable discussions.

References

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[Ang88] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[AKMW95] P. Auer, S. Kwek, W. Maass, and M. K. Warmuth. On-line learning of small threshold-circuits. In preparation.

[AL94] P. Auer and P.M Long. Simulating access to hidden information while learning. In *Proceedings of the 26th ACM Symposium on the Theory of Computation.* ACM Press, 1994. pp. 263-272.

[AW95] P. Auer and M.K. Warmuth. Tracking Shifting Disjunctions. To appear in *36th Annual Symposium on Foundations of Computer Science*, Fall 1995.

[Aue93] Peter Auer. On-line learning of rectangles in noisy environments. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 253–261, July 1993.

[BEHW89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965, October 1989.

[BGGM94] Nader H. Bshouty, Paul W. Goldberg, Sally A. Goldman, and H. David Mathias. Exact learning of discretized concepts. Technical Report WUCS-94-19, Washington University, 1994.

[BCH94] Nader H. Bshouty, Zhixiang Chen, and Steve Homer. On learning discretized geometric concepts. To appear in *35th Annual Symposium on Foundations of Computer Science*, November 1994.

[CH95] Zhixiang Chen and Steven Homer. The bounded injury priority method and the learnability of unions of rectangles. To appear in *Annals of Pure and Applied Logic*.

[CM92] Zhixiang Chen and Wolfgang Maass. On-line learning of rectangles. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 16–27. ACM Press, July 1992.

[Cov91] T. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, 1991.

[DK95] E. Dichterman and R. Khardon. A tight bound for the VC dimension of k-term DNF. Private communication.

[FGMP94] Mike Frazier, Sally Goldman, Nina Mishra, and Leonard Pitt. Learning from a consistently ignorant teacher. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, July 1994.

[Hau89] David Haussler. Learning Conjunctive Concepts in Structural Domains. *Machine Learning* 4(1):7–40, 1989.

[HKW94] D. Haussler, J. Kivinen, and M. K. Warmuth. Tight worst-case loss bounds for predicting with expert advice. Technical Report UCSC-CRL-94-36, University of California Computer Research Laboratory, Santa Cruz, CA. An extended abstract appeared in the *Proceedings of the Second European Conference, EuroCOLT'95*, Springer Verlag, Lecture Notes in Artificial Intelligence, Vol. 904, pp. 69-83.

[HS95] D. P. Helmbold and R. E. Schapire. Predicting nearly as well as the best pruning of a decision tree. To appear in the *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, Santa Cruz, July 1995.

[HSW90] D. P. Helmbold, R. Sloan and Manfred K. Warmuth. Learning Nested Differences of Intersection-Closed Concept Classes. *Machine Learning*, vol. 5, pp. 165-196, 1990.

[HW95] M. Herbster and M. K. Warmuth. Tracking shifting experts. To appear in the Proceedings of Twelfth International Conference on Machine Learning, Taos, July 1995.

[KV94] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge Massachusetts, 1994.

[KW94] Kivinen, J., Warmuth, M. K.: Exponentiated gradient versus gradient descent for linear predictors. Technical Report UCSC-CRL-94-16, Univ. of Calif. Computer Research Lab, Santa Cruz, CA, June 1994. To appear in the *Proceeding of the Twenty Seventh Annual ACM Symposium on Theory of Computing*, Las Vegas, May 1995.

[KW95] J. Kivinen and M. K. Warmuth. The Perceptron algorithm vs. Winnow: linear vs. logarith mic mistake bounds when few input variables are relevant. To appear in the *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, Santa Cruz, July 1995.

[Lit88] N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[Lit89a] N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, Technical Report UCSC-CRL-89-11, University of California Santa Cruz, 1989.

[Lit89b] N. Littlestone. From on-line to batch learning. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 269–284. Morgan Kaufmann, 1989.

[Lit91] N. Littlestone. Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages

[Lit95] N. Littlestone. Private communication.

[LW94] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[MT94] Wolfgang Maass and György Turán. Algorithms and lower bounds for on-line learning of geometrical concepts. *Machine Learning* 14, 251-269, 1994.

[MT92] Wolfgang Maass and György Turán. Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9:107–145, 1992.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

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

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[Vov90] Vovk, V.: Aggregating strategies. In *Proc. 3rd Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, San Mateo, CA, 1990.

Appendix: The improved time bound for Theorem 1 and Theorem 2

We will show a time bound of $O(d \log r)$ for prediction and for updating the representation of the hypothesis after a mistake, where r is the current number of mistakes. This is done by using for each of the $2d$ groups of virtual variables an appropriately labeled 2 - 3 tree for storing their current weights. A 2 - 3 tree [AHU74] is a tree in which each vertex which is not a leaf has 2 or 3 children, and every path from the root to a leaf is of the same

length. The internal nodes ν of a 2 - 3 tree are labeled with the largest number $L[\nu]$ by which any leaf below the leftmost child of ν is labeled, and with the largest number $M[\nu]$ by which any leaf below the second child of ν is labeled. Furthermore the numbers by which the leaves are labeled are increasing from left to right.

For each of the 2d groups of n virtual variables we employ a 2 - 3 tree that has additional labels at each node. The leafs of the tree for a group are labeled in increasing order from left to right by the left endpoints of the previously considered "blocks" of variables, together with the current weight shared by the variables of the block and the length of the block.

We would like to label each internal node ν of this 2 - 3 tree with the sum of weights of the set $V[\nu]$ of all variables that belong to blocks whose left endpoints occur as a label of some leaf below ν. However if we would do this, then it would become too time consuming to update all of these labels when the weights of all variables in a final (or initial) segment of this group of n variables are multiplied with by the factor α. Therefore we label instead the internal nodes ν of the 2 - 3 trees with *two* numbers: a *factor* and a *preliminary sum* of current weights of the variables in $V[\nu]$. The *actual* current sum of weights of the variables in $V[\nu]$ can be easily computed from these labels of internal nodes in the 2 - 3 tree by multiplying the contents of the factor labels of all nodes on the path starting at ν to the root of the tree and by multiplying the "preliminary sum" of ν by the resulting number. With this data structure one can compute the following very efficiently for any of the $2d$ groups of virtual variables: the sum of current weights for all variables in that group whose index is above (respectively below) i_0 for any given $i_0 \in \{1, \ldots, n\}$. The time needed for this operation is proportional to the *depth* of the tree, hence one needs only $O(\log r)$ time. Therefore each prediction of the learning algorithm requires altogether only $O(d \log r)$ computation steps on a RAM.

In order to update the 2 - 3 trees after a mistake, one has to multiply for each of the $2d$ groups of variables the weights of an initial or final segment of these variables with a common factor. Furthermore if the left endpoint e of that segment does not coincide with one of the endpoints of blocks that occur as labels of the respective 2 - 3 tree, then one has to create a new leaf for this endpoint e, restructure the tree so that it becomes again a 2 - 3 tree (this is necessary if the node immediately above the new leaf has already 3 children), and update the labels of nodes in this tree in accordance with the changed weights of an initial or final segment of variables in this group.

It is rather easy to see that for each of the $2d$ different 2 - 3 trees the described updating operation requires only time proportional to the depth of the tree (and hence is bounded by $O(\log r)$). If necessary, one first adds a new leaf corresponding to the new endpoint of an interval of variables in that group (use for example the procedure SEARCH of algorithm 4.4 in [AHU74]). Simultaneously one can move all "factors" that occur in the labels of nodes on the path from the root to the new leaf downwards, and compute for all internal nodes ν that lie on or immediately below this path the *actual* sum of current weights of all variables in $V[\nu]$. One updates in an analogous manner the labels of all nodes ν on the path from the root to the next leaf to the left of the new leaf (note that in general the set $V[\nu]$ changes for these nodes ν because the interval in the leaf to the left of the new leaf is shortened). Finally one restructures the resulting tree into a 2 - 3 tree (in the same way as described in [AHU74]). For that, the structure of the tree is changed only along the path from the root to the new leaf. For these internal nodes ν we have already computed the *actual* current sum of weights of all variables in $V[\nu]$, and hence we can compute appropriate new labels for

all nodes in the new 2 - 3 tree with a total number of computation steps that is proportional to the depth of the tree. ∎