Efficient Learning with Virtual Threshold Gates

Wolfgang Maass

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

Manfred K. Warmuth

Dept. of Computer and Information Sciences
University of California
Santa Cruz, CA 95064, USA
e-mail: manfred@cis.ucsc.edu

1 Introduction

We introduce a new technique for the design of efficient learning algorithms, that yields superior (and in many cases optimal) learning algorithms for a number of frequently studied learning problems in the most common formal model for online learning. If one transfers these learning algorithms fo the PAC-learning model, one gets for the same concept classes efficient PAC-learning algorithms that require fewer training-examples than the previously known algorithms.

The key idea of this new method is to use hypotheses that are computed by a "virtual threshold gate" with v new "virtual" variables, and to apply Littlestone's learning algorithm WINNOW to these v variables. WINNOW has the ability to home in very fast on those few of these v variables that are actually needed for characterizing the target concept of the original learning problem. This method is computationally feasible (in spite of the fact that in our applications the number v of "virtual" variables exceeds by far the tolerable complexity bounds) because the weights that are assigned by WINNOW to these v input variables of the virtual threshold gate are very "uniform". More precisely, for large "blocks" of virtual variables the associated weights will have the same value. Hence one can apply a simple "online compression", which allows us to compute the prediction of the current hypothesis (i.e. of the virtual threshold gate with its current weightvector) within a number of computation steps that is by far smaller than the number v of virtual variables.

We consider in this paper the most common formal model for online learning: Littlestone's "mistake bounded model" for online learning ([L88]). This model can easily be shown to be equivalent to Angluin's model [A 87] for online learning with equivalence queries (we refer to [MT 92] for a survey of these and related formal models for online learning). In order to define either of these models one fixes some arbitrary domain X and an arbitrary class $\mathcal{C}$ of subsets of X. $\mathcal{C}$ (the "concept class") is the class of all possible target concepts for the considered learning processes. At the beginning of a learning process the "environment" (also referred to as "teacher") fixes some arbitrary target concept $C_T \in \mathcal{C}$.

In Littlestone's model the learner receives at each step t of the learning process some $x \in X$, for which he then has to predict whether $x \in C_T$. Immediately afterwards the learner is told whether this prediction was correct, and one moves on to step $t+1$ of the learning process. If at step t the learner's prediction for $C_T(x)$ was incorrect, one says that the learner has made a *mistake* at step t. The goal of the learner is to minimize the worst case number of mistakes that can occur in any learning process for a target concept from $\mathcal{C}$.

Equivalently (see [A 87], [L 88]) one can describe the learner as an agent that issues a sequence of "equivalence queries" of the form "Is $H = C_T$?" for certain "hypotheses" $H \subseteq X$. Whenever $H \neq C_T$, the learner receives from the environment a counterexample $x \in (H - C_T) \cup (C_T - H)$, and he is then allowed to change his hypothesis. In this framework the learner wants to minimize the worst case number of equivalence queries that he has to ask in order to identify an arbitrary target concept from $\mathcal{C}$ (for arbitrary choices of counterexamples to his equivalence queries).

One can easily show that both models are equivalent ([L 88]). At any step of a learning process in Littlestone's mistake bounded model one can view the set of all $x \in X$ for which the learner would currently predict that $x \in C_T$ as the *current hypothesis* $H \subseteq X$ of the learner. In this way an example $x \in X$ for which the learner makes an incorrect prediction in Littlestone's model corresponds in Angluin's model to a *counterexample* for the current hypothesis of the learner. Thus the only difference between these models is apparently the fact that in Littlestone's model the learner may also get examples for which he makes correct predictions. However this is a rather superficial difference because one analysis in both models the performance of learning algorithms against a *worst case adversary.* It is obvious that a worst case adversary could always *avoid* to present any example x for which the learner would make a correct prediction (as long as $H \neq C_T$), provided that one considers only *deterministic* learning algorithms (as we do in this paper). Hence for any concept class $\mathcal{C}$ the worst case number of mistakes of the best learning algorithm for $\mathcal{C}$ in Littlestone's model is equal to the worst case number of equivalence queries that the best learning algorithm needs to identify arbitrary target concepts from $\mathcal{C}$ in Angluin's model. Because of this close relationship between both models for online learning we will employ in this paper terminology and intuitions from both models (depending on which is better suitable for a considered situation). We will also consider in this paper the somewhat more realistic variations of these models where some *noise* is permitted in the examples $\underline{x}$ (respectively in the counterexamples $\langle \underline{x}, b \rangle$, if one works in Angluin's model). Thus in the case of a domain $X \subseteq \mathbf{R}^d$ instead of the "true" example $\underline{x} \in X$ the learner receives some other vector $\underline{x}' \in X$ that differs from $\underline{x}$ in some of its coordinates (i.e. "attributes"). In this model for online learning with *attribute errors* in the examples the learner is told that the correct classification of $\underline{x}'$ is $C_T(\underline{x})$ (although in general $C_T(\underline{x}') \neq C_T(\underline{x})$). In the closely related version of this model with *classification errors* one assumes that the examples $\underline{x}$ themselves are not changed, but that the learner receives as "classification" of $\underline{x}$ the incorrect value $1 - C_T(\underline{x})$. In both versions one assumes that the respective noise is controlled by the same *adversary* that also chooses the target concept from $\mathcal{C}$ and the examples.

The results of this paper build on Littlesone's results about his online learning algorithms WINNOW1 and WINNOW2, that make predictions for a given example $\langle x_1, \ldots, x_v \rangle \in \{0,1\}^v$ depending on whether $\sum_{i=1}^{v} x_i w_i \geq \Theta$ for the current weight-vector $w_1, \ldots, w_v \in \mathbf{R}$ and a suitable fixed threshold Θ. Thus the hypotheses of these learning algorithms are halfspaces over $\{0,1\}^v$. Both WINNOW1 and WINNOW2 depend besides Θ on a certain parameter $\alpha > 1$. We assume that initially $w_1 = \ldots = w_v = 1$. Assume now that they make a mistake for some example $\langle x_1, \ldots, x_v \rangle \in \{0,1\}^v$. If $\langle x_1, \ldots, x_v \rangle \in C_T$, then both WINNOW1 and WINNOW2 replace all weights w_i for which $x_i = 1$ by the larger weight $\alpha \cdot w_i$. If $\langle x_1, \ldots, x_v \rangle \notin C_T$, then

WINNOW1 replaces all weights w_i for which $w_i = 1$ by 0, whereas WINNOW2 replaces these weights by w_i/α. Note that for both algorithms the weights w_i stays always nonnegative.

These learning algorithm performs very well if the target concepts are "short disjunctions" over $\{0,1\}^v$, i.e. concepts from the concept class $\mathcal{C}_{k,v} := \{C \subseteq \{0,1\}^v :$ there are $i_1, \ldots, i_k \in \{1, \ldots, v\}$ such that $\forall \langle x_1, \ldots, x_v \rangle \in \{0,1\}^v (\langle x_1, \ldots, x_v \rangle \in C \Leftrightarrow (x_{i_1} = 1 \vee \ldots \vee x_{i_k} = 1))\}$.
In particular:

(1.1) With $\alpha := 2$ and $\Theta := \frac{v}{2k}$ the algorithm WINNOW1 learns arbitrary target concepts from $\mathcal{C}_{k,v}$ with at most $2k(1 + \log v/k)$ mistakes.

(1.2) With $\alpha := 1.5$ and $\Theta := v$ the algorithm WINNOW2 makes at most $14k \ln v + 4z + 8$ mistakes in any learning process for a target concept from $\mathcal{C}_{k,v}$ with a total of at most z attribute errors in the examples, and at most $14k \ln v + 4kq + 8$ mistakes in any learning process for a target concept from $\mathcal{C}_{k,v}$ with a total of at most q classification errors in the examples (note that each classification error can also be viewed as up to k attribute errors in the respective example).

Thus the algorithm WINNOW1 has somewhat smaller constant factors in its mistake bound, whereas WINNOW2 tolerates a certain amount of noise in the examples. Both algorithm home in very fast on the $\leq k$ attributes that are *relevant* for the target concept form $\mathcal{C}_{k,v}$, since they depend only on the *logarithm* of the total number of attributes. This is essential for the results of this paper where we will reduce the given learning problems to applications of WINNOW with an exponential number v of attributes. The other property of WINNOW that we exploit in this approach is that it changes the weights in a very *uniform* manner: all w_i with $x_i = 1$ are multiplied with the *same* factor.

2 Efficient Online Learning of Simple Geometrical Objects in Unbounded Dimension

We first consider the problem of efficient online learning of a single axis-parallel rectangle over the domain

$$X_n^d := \{1, \ldots, n\}^d,$$

i.e. of arbitrary target concepts from the concept class

$$BOX_n^d := \left\{ \prod_{i=1}^{d} \{a_i, a_i + 1, \ldots, b_i\} : 1 \leq a_i \leq b_i \leq n \text{ for } i = 1, \ldots, d \right\} \cup \{\emptyset\}.$$

The complement $\overline{C} := X_n^d - C$ of any concept $C = \prod_{i=1}^{d} \{a_i, \ldots, b_i\} \in BOX_n^d$ 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 $v := 2dn$ "virtual variables" $u_{i,c}^{<}$ and $u_{i,c}^{>}$ for $i = 1, \ldots, d$ and $c = 1, \ldots, n$. These v new variables are associated with the $2dn$ halfspace $H_{i,c}^{<} := \{\underline{x} \in X_n^d : (\underline{x})_i < c\}$ and $H_{i,c}^{>} := \{\underline{x} \in X_n^d : (\underline{x})_i > c\}$ for $i = 1, \ldots, d$ and $c = 1, \ldots, n$. By the preceding observation any $\overline{C}$ for $C \in BOX_n^d$ can be represented as the OR of $2d$ of these new variables. Hence we can exploit the advantages of WINNOW which is able to focus very fast the attention on these few "relevant" variables.

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 previously described set of $v := 2dn$ variables. Instead, we execute a "compressed version" of WINNOW that keeps 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 $\log n$. Thus we can prove the following result.

Theorem 1: *There exists an online learning algorithm for BOX_n^d that makes at most $O(d \cdot \log n)$ mistakes, and which uses at most $O(d(\log d + \log\log n))$ computation steps to update its hypothesis after a mistake, or to decide whether $\underline{y} \in H$ for its current hypothesis $H \subseteq \{1, \ldots, n\}^d$ and any $\underline{y} \in \{1, \ldots, n\}^d$.*

This learning algorithm is optimal in a rather strong sense: any online learning algorithm for BOX_n^d (even a learning algorithm that uses arbitrary sets $H \subseteq \{1, \ldots, n\}^d$ as hypotheses, that may use also membership queries, and that uses an arbitrary number of computation steps) has to use $\Omega(d \cdot \log n)$ queries for some learning process.

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 online 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 online 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 the "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 $u^<_{i,c}(u^>_{i,c})$ equal to 1 if $\underline{y} \in H^<_{i,c}(H^>_{i,c})$, 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 $C_T \in BOX^d_n$ 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 $1 \leq a_i \leq b_i \leq n$ 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 online learning $\overline{C}_T$ to the problem of online 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 $i = 1,\ldots,d, c = 1,\ldots,n$. 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 online learning algorithm for the complements $\overline{C}_T$ of rectangles $C_T \in BOX^d_n$ 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 $\Theta := v/2d = n$, then one gets for this learning problem a mistake bound of $\alpha k(1 + \log_\alpha v/k) = 4d(1 + \log n)$. With $\alpha := e$ and $\Theta := \frac{n}{e}$ one arrives at a mistake bound of $3.77d(1.45 + \log n)$, which is slightly smaller for most values of d and n (see [L 88]).

With regard to the *lower* bound for the number of queries it had already been pointed out in [CM] that *any* online learning algorithm for BOX^d_n with arbitrary equivalence queries and arbitrary computational power can be forced by the "adversary" to make $\Omega(d \cdot \log n)$ mistakes, since one can construct for BOX^d_n a "mistake tree" in which every path from a root to a leaf has length $\Omega(d \cdot \log n)$. This lower bound argument exploits an earlier result of Littlestone [L 88], who had shown that such a mistake tree can be used as an adversary strategy, which forces any learner who uses only equivalence queries to make $\Omega(d \cdot \log n)$ queries. A mistake tree (also called adversary tree in [MT 92]) is a binary branching tree in which each internal node is labeled by some point $\underline{x}$ from the domain X, and where from each internal node exactly 2 edges emerge, that are labeled by 0 and 1, respectively. Furthermore each leaf ℓ of such tree is labeled by some $C \in \mathcal{C}$ such that $C(\underline{x}) = b$ for each $\underline{x}$ such that the path from the root to ℓ passes some node with label $\underline{x}$ and leaves that node via an edge with label b.

In order to construct a mistake tree for BOX^d_n in which every path from the root to a leaf has length $\geq 2d \lfloor \log \lfloor n/2 \rfloor \rfloor = \Omega(d \cdot \log n)$, one employs only those

$C = \prod_{i=1}^{d} \{a_i, \ldots, b_i\} \in BOX_n^d$ as labels of leafs for which $a_i \in \{1, \ldots, \lfloor n/2 \rfloor\}$ and $b_i \in \{\lfloor n/2 \rfloor, \ldots, n\}, i = 1, \ldots, d$. The labels $\underline{x}$ of the internal nodes are chosen in such a way that the resulting sequence of questions "$\underline{x} \in C$?" along any path implements a sequence of successive binary searches for the coordinates $a_1, \ldots, a_d, b_1, \ldots, b_d$ of the cornerpoints of a concept $C \in BOX_n^d$.

Finally, it has been shown by Auer and Long [AL] that for any concept class $\mathcal{C}$ the number of queries of an optimal online learning algorithm can be reduced at best by a constant factor if it uses also membership queries besides arbitrary equivalence queries.

The computation time of our learning algorithm for BOX_n^d can be estimated as follows. After s counterexamples each group of variables $u_{i,1}^<, \ldots, u_{i,n}^<$ (resp. $u_{i,1}^>, \ldots, u_{i,n}^>$) consists of up to $s+1$ "intervals" $u_{i,k}^<, u_{i,k+1}^<, \ldots, u_{i,\ell}^<$ (respectively $u_{i,k}^>, u_{i,k+1}^>, \ldots, u_{i,\ell}^>$) 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 $s+1$ intervals, together with the current weight of the associated virtual variables.

By the preceding analysis we can assume that $s = O(d \log n)$. Hence without use of a more sophisticated datastructure 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.

One can reduce the computation time for both of these tasks to $O(d \cdot \log(d \log n))$ 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 (see [AHU]) is a tree in which each vertex which is not a leaf has 2 or 3 sons, 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 $L[\nu]$ that occurs in the label of a leaf below the leftmost son of ν, and with the largest number $M[\nu]$ that occurs in the label of a leaf below the second son of ν. Furthermore certain numbers occur in increasing order from left to right as labesls of its leafs.

We employ here for each of the 2d groups of virtual variables a 2 - 3 tree that has additional labels at each node. The leafs of that tree are labeled in increasing order from left to right by the left endpoints of the previously considered "intervals" of variables, together with the common current weight of the variables in that interval (whose left endpoint is recorded at the leaf) and the number of variables in it.

In principle we would like to label in addition 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 intervals 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 these n variables are multiplied with a common factor. This would in general require to update the labels of a large portion of all nodes in that 2 - 3 tree. Therefore we label instead the internal nodes ν of the 2 - 3 trees with *two* numbers: a certain *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 "preliminary sum" which occurs in the label of ν with all "factors" which are stored at ν or at any nodes above ν.

With this datastructure one can compute very efficiently for any of the $2d$ groups of virtual variables and any given $i_0 \in \{1, \ldots, n\}$ the sum of current weights for all variables in that group whose index is above (respectively below) i_0. The number of computation steps that one needs for that is proportional to the *depth* of that tree, hence one needs only $O(\log(d \log n))$ steps. Therefore each prediction of the learning algorithm requires altogether only $O(d \cdot \log(d \log n))$ computation steps on a RAM.

In order to update these $2d$ different 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 right (respectively left) endpoint e of that segment does not coincide with one of the endpoints of intervals that occur as labels of the respective 2 - 3 tree, 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 sons), 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 a number of computation steps that is proportional to the depth of that tree (and hence is bounded by $O(\log(d \log n))$). 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 [AHU]). 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 its last "internal" is shortened by inserting the endpoint that occurs as label of the new leaf). Finally one restructures the resulting tree into a 2 - 3 tree (in the same way as described in [AHU]). 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. ∎

In the next result we consider the somewhat more realistic formal model for online learning, where one allows that some of the examples that are given to the learner are erroneous. If the learner receives an example $\langle \underline{x}, b\rangle$ with $\underline{x} \in \{1,\ldots,n\}^d$ and $\{0,1\} \ni b \neq C_T(\underline{x})$, one says that this example contains a *classification error*, or alternatively that it contains ℓ *attribute errors* if ℓ is the minimum number of coordinates of $\underline{x}$ that have to be changed so that $b = C_T(\underline{x}')$ for the resulting vector $\underline{x}' \in \{1,\ldots,n\}^d$.

Theorem 2: *There exists a noise-robust online learning algorithm for BOX_n^d that makes in any learning process where the examples contain altogether a attribute errors (or q classification errors) at most $O(a + d\log n)$ (respectively $O(d(q + \log n))$) mistakes, and which needs at most $O(d\log(a + d\log n))$ (respectively $O(d\log(d(q + \log n)))$) computation steps to update its hypothesis after a mistake, or to decide whether $\underline{y} \in H$ for its current hypothesis H and any $\underline{y} \in \{1,\ldots,n\}^d$.*

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 1, 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 in the transformed learning process). ∎

It is well-known that there exists for WINNOW a trade-off between noise-tolerance and computation time: it can tolerate a larger percentage of classification noise than that given by (1.2), at the expense of a somewhat 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 from 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 $K \in \boldsymbol{N}$ be some arbitrary parameter. Then there exists an online learning algorithm for BOX_n^d that makes at most*

$$14\left\lceil\frac{2d}{K}\right\rceil \cdot \ln\left(\left\lceil\frac{2d}{K}\right\rceil \cdot n^K\right) + 4\left\lceil\frac{2d}{K}\right\rceil \cdot q + 8$$

mistakes in any learning process with $\leq q$ classification errors in the examples. This *learning algorithm requires*

$$O\Big(\left\lceil \frac{2d}{K} \right\rceil \cdot \left(14 \left\lceil \frac{2d}{K} \right\rceil \cdot \ln \left(\left\lceil \frac{2d}{K} \right\rceil \cdot n^K \right) + 4 \left\lceil \frac{2d}{K} \right\rceil \cdot q + 9 \right)^K \Big)$$

computation steps for each prediction and each update of its hypothesis.

Proof: Consider the $2d$ groups of virtual variables that were discussed in the proof of Theorem 1. We partition these $2d$ gropus into $\ell := \lceil \frac{2d}{K} \rceil$ classes $B_1, \ldots, B_\ell$ that each consist of K or less groups of virtual variables. For the sake of simplicity we assume in the following that each of these classes consists of exactly K groups of virtual variables. We then create new "virtual variables" of types $B_1, \ldots, B_\ell$. For each $B_j (j \in \{1, \ldots, \ell\})$ the variables of type B_j represent arbitrary disjunctions of K variables with one arbitrary variable chosen from each of the K groups of virtual variables that belong to class B_j. Hence there are n^K variables of type B_j.

Thus we have created altogether $\ell \cdot n^K$ new virtual variables, and each complement of a target concept $C_T \in BOX_n^d$ can be represented as a disjunction of ℓ of these new variables. We then apply WINNOW2 with $\Theta := \ell \cdot n^K$ and $\alpha := 3/2$ in order to learn arbitrary complements of concepts from BOX_n^d with regard to this new representation. According to (1.2) this yields for any learning process with at most q incorrectly classified examples a mistake bound of $14\ell \ln(\ell \cdot n^K) + 4\ell q + 8$.

In order to compute each prediction of WINNOW2 in an efficient manner, one exploits that for each of the ℓ types B_j of variables the variables of this type can be identified with points in the K-dimensional space $\{1, \ldots, n\}^K$. Furhter at each mistake for some example $\langle \underline{y}, b \rangle$ the set of variables of type B_j whose weight is increased (respectively decreased), forms a union of K orthogonal halfspaces. Hence after r mistakes the n^K variables of type B_j (viewed as points in $\{1, \ldots, n\}^K$) have been partitioned by these unions of halfspaces into up to $(r+1)^K$ axis-parallel "rectangles" of variables so that all varables 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 datastructure requires after r mistakes at most $O((r+1)^K)$ computation steps. ∎

The preceding results can be extended to the learning of target concepts from

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

The following theorem shows that for $k = O(1)$ the complexity bounds of a learning algorithm for $\bigcup_{\leq k} BOX_n^d$ that applies WINNOW to a virtual threshold gate are polynomial in d and $\log n$.

Theorem 4: *One can learn arbitrary target concepts from $\bigcup_{\leq k} BOX_n^d$ over the domain $\{1, \ldots, n\}^d$ by a noise-robust online learning algorithm that makes at most $O((2d)^k(q + \ln((2dn)^k)))$ mistakes in any learning process with up to q classification errors in the examples. This algorithm uses (for some constant $K > 0$) at most $(K \cdot d)^k \cdot (2d)^k(q + k\ln(2dn) + 1)^k$ computation steps for a prediction, or for an update of 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 $C \in \bigcup_{\leq k} BOX_n^d$ can obviously be represented as the union of k intersections of complements of such halfspaces, i.e. 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 $C \in \bigcup_{\leq k} BOX_n^d$ 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 type 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 $C_T \in \bigcup_{\leq k} BOX_n^d$ to the learning of a disjunction of at most $(2d)^k$ of these $(2dn)^k$ virtual varaibles. 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 $\bigcup_{\leq k} BOX_n^d$ we observe that the $(2dn)^k$ virtual variables naturally fall into $(2d)^k$ groups of variables of the same "type". 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 such that for each $j \in \{1, \ldots, k\}$ there exist $i \in \{1, \ldots, d\}$ and $c, c' \in \{1, \ldots, n\}$ such that $H_{(j)} = H^<_{i,c}$ and $\tilde{H}_{(j)} = H^<_{i,\tilde{c}}$, or $H_{(j)} = H^>_{i,c}$ and $\tilde{H}_{(j)} = H^>_{i,\tilde{c}}$.

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 (assuming that the variables of this type represent intersections $\bigcap_{j=1}^{k} H(j)$ of axis-parallel halfspaces $H(j)$), these are "rectangles" with regard to the k dimensions

$i \in \{1,\ldots,d\}$ such that $H_j = H_{i,c}^{<}$ or $H_j = H_{i,c}^{>}$ for some $j \in \{1,\ldots,k\}$). The sum of the weights of all variables in any such k-dimensional "rectangle" can be computed by computing in k computation steps its "volume", and by multiplying this volume with the common weight of these variables. According to (1.2) WINNOW2 makes at most $s = O((2d)^k(q + \log((2dn)^k)))$ mistakes in any learning process that results through the described transformation from a learning process with the complement of some $C_T \in \bigcup_{\leq k} BOX_n^d$ as target concept and at most q classification errors in the examples. Hence each prediction of this learning algorithm and each update of its hypothesis after a mistake requires at most $O((2d)^k \cdot k \cdot (O((2d)^k(q+\log((2dn)^k)) + 1))^k)$ computation steps on a RAM. ∎

The following result shows that one can easily extend the previously discussed algorithms for online learning of orthogonal rectangles and unions of orthogonal rectangles to yield efficient algorithms for online learning of arbitrary unions of $O(1)$ intersections of halfspaces whose normal-vectors belong to some *arbitrary* given set $\mathcal{D}$ of "directions".

Theorem 5: *Assume that an arbitrary finite set $\mathcal{D} \subseteq \boldsymbol{R}^d$ of "directions" is given.*

$$\text{Set } \mathcal{C}_{\mathcal{D},k} := \{C_1 \cup \ldots \cup C_{k'} \ : \ k' \leq k \text{ and each } C_j (j = 1,\ldots,k') \text{ is an arbitrary intersection of halfspaces of the form } \{\underline{x} \in \{1,\ldots,n\}^d : \underline{x}\cdot\underline{a} \geq \Theta\} \text{ or } \{\underline{x} \in \{1,\ldots,n\}^d : \underline{x}\cdot\underline{a} \leq \Theta\} \text{ with arbitrary } \Theta \in \boldsymbol{R} \text{ and } \underline{a} \in \mathcal{D}\}.$$

Then there exists a noise-robust online learning algorithm for $\mathcal{C}_{\mathcal{D},k}$ that makes at most $O((2|\mathcal{D}|)^k(q + \ln(|\mathcal{D}|\cdot n^d)))$ mistakes in any learning process with at most q classification errors in the examples. For $k = O(1)$ the number of computation steps that are used by this algorithm for a prediction, or for updating its hypothesis after a mistake, is polynomial in d, $|\mathcal{D}|$ and $\log n$.

Proof: For the case $k = 1$ it suffieces to introduce for each direction $\underline{a} \in \mathcal{D}$ and each $\Theta \in \mathbf{R}$ such that $\underline{x}\cdot\underline{a} = \Theta$ for some $\underline{x} \in \{1,\ldots,n\}^d$ two virtual variables $u_{\underline{a},\Theta}^{\leq}$ and $u_{\underline{a},\Theta}^{\geq}$. Analogously as in the proof of Theorem 1 one can reduce the problem of learning the complement of an arbitrary target concept from $\mathcal{C}_{\mathcal{D},1}$ to the problem of learning a disjunction of up to $2|\mathcal{D}|$ of these $\leq 2|\mathcal{D}|\cdot n^d$ virtual variables. For that purpose one defines a transformation $\{1,\ldots,n\}^d \ni \underline{y} \mapsto \underline{u}(\underline{y}) \in \{0,1\}^{2|\mathcal{D}|\cdot n^d}$ by setting $u_{\underline{a},\Theta}^{\leq}(\underline{y}) = 1$ if and only if $\underline{a}\cdot\underline{y} \leq \Theta$ (respectivly $u_{\underline{a},\Theta}^{\geq}(\underline{y}) = 1$ if and only if $\underline{a}\cdot\underline{y} \geq \Theta$). One then applies WINNOW2 to this transformed learning problem.

For the case $k > 1$ one introduces analogously as in the proof of Theorem 4 a separate virtual variable for any intersection of k halfspaces of the type considered above. In this way one can reduce the problem of learning the complement of an arbitrary target concept from $\mathcal{C}_{\mathcal{D},k}$ to the problem of learning a disjunction of up to $(2|\mathcal{D}|)^k$ of the $2\cdot|\mathcal{D}|\cdot n^d$ virtual variables. One then applies WINNOW2 in the same way as in the proof of Theorem 4. We assume here that in a separate preprocessing phase one has computed for each $\underline{a} \in \mathcal{D}$ the order-number of each element of the linearly ordered set $\{\underline{x} \cdot \underline{a} \;:\; \underline{x} \in \{1, \ldots, n\}^d\}$. One can then compute in $O(k)$ steps in the same way as in the proof of Theorem 4 the "volume" of each k-dimensional "rectangle" of virtual variables. ∎

3 Efficient Online Learning of Complex Geometrical Objects in a Fixed Dimension

We first show that with the same method as in the preceding section one can also design an online learning algorithm for $\bigcup_{\leq k} BOX_n^d$ whose complexity bounds are polynomial in k and $\log n$ for $d = O(1)$.

Theorem 6: *There exists an online learning algorithm for $\bigcup_{\leq k} BOX_n^d$ that uses $O(k(q + d \log n))$ equivalence queries in any learning process with at most q classification errors in the examples, and which uses at most $(O(kd \ln n + kq))^{2d}$ computation steps for each prediction and update of its hypothesis. The number of queries of this learning algorithm is optimal in a rather strong sense: any online learning algorithm for $\bigcup_{\leq k} BOX_n^d$ (even with arbitrary computational power, equivalence queries with arbitrary sets $H \subseteq \{1, \ldots, n\}^d$ as hypotheses, and with membership queries) has to make $\Omega(kd \log n)$ queries in some learning process for $\bigcup_{\leq k} BOX_n^d$ (even without classification errors in the examples).*

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 $a_i \leq b_i$ for $i = 1, \ldots, d$ and $\underline{y} \in \prod_{i=1}^{d} \{a_i, \ldots, b_i\}$.

Obviously any target concept $C_T \in \bigcup_{\leq k} BOX_n^d$ can be represented as a disjunction of up to k of these virtual variables. According to (1.2) at most $O(k(q + d \log n))$ mistakes can occur when one applies WINNOW2 to the transformed learning problem with n^{2d} virtual variables. 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 $\{1, \ldots, n\}^{2d}$.

In order to prove the claimed *lower bound* one constructs (analogously as in the proof of Theorem 1) a "mistake tree" in which every path from the root to a leaf has length $\geq 2k \cdot \log\lfloor \frac{n}{2k} \rfloor + 2(d-1)k \log\lfloor \frac{n}{2} \rfloor = \Omega(kd \log n)$. In this construction one reserves in the first dimension a separate interval for each of the $2k$ first coordinates of cornerpoints of the k disjoint boxes that occur as labels of the leafs of this mistake tree. It is useful to make these k boxes disjoint to avoid undesired "interactions" between the binary searches for the coordinates of their cornerpoints. The first $2k \log\lfloor \frac{n}{2k} \rfloor$ levels of this mistake tree implement $2k$ sequential "binary searches" for the $2k$ first coordinates of cornerpoints. In the subsequent $2(d-1)k \log\lfloor \frac{n}{2} \rfloor$ levels one implements $2(d-1)k$ binary searches for the remaining $d-1$ coordinates of the cornerpoints. ∎

Remains to be added to this first draft of our paper

- definitions of the concept classes that are considered and a discussion which results will be discussed in later sections
- discussions of related results (best as "Remarks" after our Theorems, or at the end of sections for several results together)
- learning of threshold circuits of depth 2 with inputs from $\{1, \ldots, n\}^d$ for fixed d, allowing noise in the examples (should be added at the end of section 3)
- ? learning of neural nets of depth 2 with real valued output (using results of Kivinen/Warmuth)
- applications of our results for online learning over a discrete space to PAC-learning over a continuous space (i.e. to a given sample of m points from $\mathbf{R}^d$)
- application of our noise-robustness in online learning to PAC-learning with malicious noise.
- Conclusions
- References