On-line Learning of Rectangles and Unions of Rectangles

Zhixiang Chen * Wolfgang Maass **

Abstract

We design efficient algorithms for on-line learning of axis-parallel rectangles (and for the union of two such rectangles) in the common model for on-line learning with equivalence queries. With regard to the learning of rectangles in arbitrary dimensions d we solve the following open problem:

Is there an algorithm for on-line learning of rectangles $\prod_{i=1}^{d} \{a_i, a_i+1, \ldots, b_i\}$ over a discrete domain $\{1, \ldots, n\}^d$ whose error bound is polylogarithmic in the size n^d of the domain (i.e. polynomial in d and $\log n$)?

We give a positive solution by introducing a new design technique that appears to be of some interest on its own. The new learning algorithm for rectangles consists of $2d$ separate search strategies that search for the parameters $a_1, b_1, \ldots, a_d, b_d$ of the target rectangle. A learning algorithm with this type of modular design tends to fail because of the well known "credit assignment problem": Which of the $2d$ local search strategies should be "blamed" when the global algorithm makes an error? We propose here a rather radical solution to this problem: *each* local search strategy that is possibly involved in an error of the global algorithm will be blamed. With this radical solution it is unavoidable that frequently local search strategies will be blamed incorrectly. We overcome this difficulty by employing local search strategies ("error tolerant binary search") that are able to *tolerate* such incorrect credit assignments. The structure of this learning algorithm is reminiscent of "finite injury priority constructions" in recursive function theory.

Section 4 contains another application of this design technique: an algorithm for learning the union of two rectangles in the plane.

*Department of Computer Science, Boston University, MA 02215, USA. E-mail: zchen@cs.bu.edu.

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

1 Introduction

In our first result we consider the concept class

$$BOX_n^d := \{\phi\} \cup \left\{ \prod_{i=1}^{d} \{a_i, \ldots, b_i\} \;\middle|\; 1 \le a_i \le b_i \le n \text{ for } i = 1, \ldots, d \right\}$$

over the domain $\{1, \ldots, n\}^d$. We will refer to the elements of BOX_n^d as *boxes*, or equivalently as *rectangles*.

Our learning model is the standard model for on-line learning (see [A], [L], [MTc]). A *learning process* for a concept class $\mathcal{C}$ over a domain X is viewed as a dialogue between a *learner* A and the *environment*. The goal of the learner A is to "learn" an unknown target concept $C_T \in \mathcal{C}$ that has been fixed by the environment. In order to gain information about C_T the learner proposes hypotheses H from a fixed hypothesis space $\mathcal{H}$ with $\mathcal{C} \subseteq \mathcal{H} \subseteq 2^X$.

Whenever $H \neq C_T$ for the proposed hypothesis H, the environment responds with some *counterexample* (CE) $g \in H \Delta C_T := (C_T - H) \cup (H - C_T)$. g is called a *positive counterexample* (PCE) if $g \in C_T - H$, and g is called a *negative counterexample* (NCE) if $g \in H - C_T$. Each new hypothesis H_{s+1} of the learner (resp. learning algorithm) A may depend on the earlier hypotheses $H_1, \ldots, H_s$ and the given counterexamples $g_j \in H_j \Delta C_T$ for $j = 1, \ldots, s$.

One defines the resulting *learning complexity of a* *learning algorithm* A by

$$LC(A) := \max\{s \in \mathbf{N} \mid \text{there is some } C_T \in \mathcal{C} \text{ and a sequence } g_1, \ldots, g_{s-1} \text{ of counterexamples to the hypotheses } H_1, \ldots, H_{s-1} \text{ of the learner } A \text{ such that } H_s \neq C_T\}.$$

The *learning complexity of concept class* $\mathcal{C}$ *with hypothesis space* $\mathcal{H}$ is defined by

$$LC^{\mathcal{H}}(\mathcal{C}) := \min\{LC(A) \mid A \text{ is a learning algorithm for } \mathcal{C} \text{ with hypothesis space } \mathcal{H}\}.$$

One sets $LC(\mathcal{C}) := LC^{\mathcal{C}}(\mathcal{C})$ and $LC - ARB(\mathcal{C}) := LC^{2^X}(\mathcal{C})$.

This learning model is equivalent to Littlestone's model [L] for "mistake bounded on-line learning". In Littlestone's model, which is somewhat closer to realistic learning situations, the learner proposes at each step s some hypothesis $H_s \in \mathcal{H}$. He then uses this hypothesis H_s to "predict" the label y_s of the example $\langle x_s, y_s\rangle \in X \times \{0,1\}$ (with $y_s = C_T(x_s)$) that is given without the label y_s to the learner at step s. Whenever this prediction is incorrect, one says that the learner has made a mistake at step s. The goal of the learner is to minimize the total number of mistakes that he makes. It is easy to show (see [L]) that the worst case mistake bound of the best learning algorithm in this model is equal to $LC^{\mathcal{H}}(\mathcal{C})$. This holds no matter whether the learner is allowed to revise his hypothesis at every step, or only at those steps where he has made a mistake.

It is known that $LC(BOX_n^d) \geq LC - ARB(BOX_n^d) = \Theta(d \log n)$. The upper bound $O(d \log n)$ for $LC - ARB(BOX_n^d)$ follows by considering the HALVING-algorithm (see [A], [L], [MTc]). The lower bound $\Omega(d \log n)$ is shown by constructing a decision tree for BOX_n^d in which every leaf has depth $\Omega(d \log n)$. This is sufficient by a result of Littlestone [L] (see also [MTc]).

The HALVING-algorithm uses arbitrary subsets of the domain as hypotheses. With regard to learning algorithms for BOX_n^d that use computationally feasible hypotheses there exist two quite different approaches. Both of these algorithms use hypotheses from BOX_n^d. There is a learning algorithm B with $LC(B) = O(d \cdot n)$ that issues as its next hypothesis always the smallest $C \in BOX_n^d$ that is consistent with all preceding counterexamples. This algorithm is frequently considered in the special case $n = 2$ where the concepts $C \in BOX_2^d$ correspond to monomials over d boolean variables (see the algorithm for the complementary class 1-CNF in [V]).

It is less trivial (even for $d = 2$) to design a learning algorithm D for BOX_n^d with computationally feasible hypotheses such that $LC(D) = O(f(d) \log n)$ for some function $f : \mathbf{N} \to \mathbf{N}$. An algorithm D of this type (which uses hypotheses from BOX_n^d) was exhibited in [MTa], [MTb]. However this algorithm D learns separately each of the 2^d corners of the target concept, and hence $LC(D)$ is exponential in d (i.e. $f(d) \geq 2^d$).

The question whether the advantageous features of both learning algorithms B and D can be combined in a single algorithm S with $LC(S) \leq \text{poly}(d, \log n)$ was first brought to our attention by David Haussler ([H], see also [MTb]).

A learning algorithm S which achieves this performance is exhibited in section 3 of this paper. It proceeds in a completely different way than the two previously described learning algorithms for BOX_n^d. We describe the main component of the new algorithm in section 2.

The main ingredient of this learning algorithm is a novel solution of the "credit assignment problem". The *credit assignment problem* may be defined as "the problem of assigning credit or blame to the individual decisions that led to some overall result" (Cohen and Feigenbaum [CF]). Obviously this problem is ubiquitous not just in Artificial Intelligence, but also in the study of adaptive neural networks, where credit or blame for the overall performance of the network has to be distributed to the individual components of the network.

The credit assignment problem in the case of online learning of rectangles is the following. When the learner receives a negative counterexample $\langle x_1, \ldots, x_d\rangle$ to his current hypothesis $\prod_{i=1}^{d} \{a_i, \ldots, b_i\}$, it is clear that the learner has to change at least one of the intervals $\{a_i, \ldots, b_i\}$ so that it no longer contains x_i. But it is not clear *which* of the intervals $\{a_i, \ldots, b_i\}$ should be changed.

Our new learning algorithm for rectangles consists of $2d$ separate search strategies that search for the $2d$ endpoints $a_1^T, b_1^T, \ldots, a_d^T, b_d^T$ of the d intervals $\{a_i^T, \ldots, b_i^T\}$ of the target-rectangle $\prod_{i=1}^{d} \{a_i^T, \ldots, b_i^T\}$. The main problem is which of the $2d$ local search strategies should be "blamed" when the global algorithm makes an error? We propose here a rather radical solution to this credit assignment problem: *each* local search strategy that is possibly involved in an error of the global algorithm will be blamed. With this radical solution it is unavoidable that frequently local search strategies will be blamed incorrectly. We overcome this difficulty by employing local search strategies ("error tolerant binary search") that are able to *tolerate* such incorrect credit assignments.

Our solution of the credit assignment problem is quite reminiscent of a famous construction method in recursive function theory, the socalled *finite injury priority construction* (see Soare [S]). This linkage is of some methodological interest insofar as priority arguments with injuries are the dominant design technique in recursive function theory, but so far there are hardly any applications of this technique for the design of concrete algorithms in computer science.

Section 4 contains another application of this design technique: an algorithm for learning the union of two rectangles in the plane. We assume here that the learner knows already that the top left corner of the domain is contained in one rectangle, and the bottom right corner in the other. Nevertheless this learning problem is substantially more complicated than the preceding one: The obvious local search procedures that search for the lengths of the sides of the two rectangles are likely to get not only *false negative counterexamples* (as in the preceding learning problem), but also *false positive counterexamples.* This complication arises from the fact that in general the learner does not know to which one of the two rectangles of C_T a positive counterexample belongs. Nevertheless one can construct for this learning problem an efficient learning algorithm whose learning complexity is asymptotically optimal. Again this algorithm consists of suitable versions of binary search as modules, which will tolerate certain incorrect credit assignments.

This positive result for learning the union of two rectangles provides a contrast to earlier results about efficiently learnable concept classes $\mathcal{C}$ such as halfplanes over $\{1,\ldots,n\}^2$, or monomials, for which one has shown that $\mathcal{U}-2-\mathcal{C} := \{C_1 \cup C_2 \mid C_1, C_2 \in \mathcal{C}\}$ is not efficiently learnable (see [MTb], [PV]).

2 An Algorithm for Binary Search that Tolerates One-sided Errors

In this section we consider an extension of the notion of a "negative counterexample", and along with it an extension of the previously described learning model.

Assume $C_T \in \mathcal{C}$ is the target concept and H_s is the current hypothesis of the learner. The environment may respond in the extended model with a positive counterexample ("PCE") $g \in C_T - H_s$, with a *true negative counterexample* ("true NCE") $g \in H_s - C_T$, or with a *false negative counterexample* ("false NCE") $g \in H_s \cap C_T$. Note that the environment is allowed to respond with a false NCE even if $H_s = C_T$. We extend the notion of a negative counterexample (NCE) so that it subsumes both true and false NCE's. The environment is not required to tell the learner to which of these categories a counterexample g belongs.

We define a binary search algorithm TBS_n (the "T" stands for error-**t**olerant) for learning the "head" h of a halfinterval $\{1,\ldots,h\} \subseteq \{1,\ldots,n\}$ in this extended learning model. The new algorithm S for learning rectangles $C_T = \prod_{i=1}^{d} \{a_i,\ldots,b_i\} \in BOX_n^d$ (see section 3) will consist of $2d$ separate copies of the here defined error-tolerant binary search algorithm TBS: in each dimension i it uses separate copies of TBS and its symmetric counterpart TBS^* for learning the "head" b_i and the "tail" a_i of the interval $\{a_i,\ldots,b_i\}$. Although this learning algorithm S for BOX_n^d will receive only true counterexamples, the individual binary search procedures may also receive *false* negative counterexamples. This is a consequence of our quite radical solution to the associated "credit assignment problem", where we blame *each* of the $2d$ subroutines for binary search for any error of the learning algorithm S. In particular a true NCE for S will result in a true NCE for at least one subroutine and false NCE's for up to $d-1$ other subroutines.

In this section we consider the concept class

$$HEAD_n := \{\{1,\ldots,j\} \mid \quad j \in \{1,\ldots,n\}\}$$

over the domain $\{1,2,\ldots,n\}$.

At the beginning of each step r of a learning process in the extended learning model the learner issues a hypothesis $H_r := \{1,\ldots,h_r\} \in HEAD_n$. If $H_r \neq C_T$, then the learner will receive at step r the counterexample $g_r \in \{1,\ldots,n\}$. We set

$$\begin{aligned}
p_s &:= \max\ (\{1\} \cup \{g_r \mid 1 \le r \le s \text{ and } \\
&\qquad g_r \text{ was a PCE}\}) \\
n_s &:= \min\ (\{n+1\} \cup \{g_r \mid 1 \le r \le s, g_r > p_s, \\
&\qquad \text{and } g_r \text{ was a (true or false) NCE}\}) \\
n_s^{true} &:= \min\ (\{n+1\} \cup \{g_r \mid 1 \le r \le s \text{ and } \\
&\qquad g_r \text{ was a true NCE}\}).
\end{aligned}$$

The interpretation of these parameters is quite obvious. p_s is the largest PCE received by step s, and n_s^{true} is the smallest true NCE received by step s. Hence it is clear that the endpoint h of the target concept

$H_T = \{1, \ldots, h\} \in HEAD_n$ lies in the "undetermined interval" $\{p_s, \ldots, n_s^{\text{true}} - 1\}$. Unfortunately the learner does in general not know the value of n_s^{true} at step s. His best possible approximation to n_s^{true} is provided by the parameter n_s, which is the minimum of all those NCE's received by step s which have not yet been shown to be false NCE's by the end of step s.

The following search algorithm TBS_n (where TBS stands for error-tolerant binary search) provides the basic module for our learning algorithm for rectangles.

Definition of the binary search algorithm TBS_n for learning $HEAD_n$ in the extended learning model:

The algorithm TBS_n issues at step s the hypothesis $\{1, \ldots, h_s\}$.
Set $h_1 := 1$.
For $s \geq 1$ set $h_{s+1} := h_s$ if g_s is a NCE and $g_s \leq p_s$.
[If $g_s \leq p_s$ it is clear that g_s is a *false* NCE, hence it can safely be ignored.]
Else, we define

$$h_{s+1} := \begin{cases} \min(\{n_s - 1\} \cup \{h_r \mid 1 \leq r \leq s \text{ and } p_s \leq h_r < n_s\}) & \text{, if } g_s \text{ is a PCE.} \\ p_s + \left\lfloor \frac{n_s - p_s}{2} \right\rfloor & \text{, if } g_s \text{ is a (true or false) NCE.} \end{cases}$$

In this algorithm TBS_n the hypothesis $\{1, \ldots, h_{s+1}\}$ at step $s+1$ is defined as in the usual binary search procedure in the case where g_s is a NCE which is not obviously false at step s (i.e. $g_s > p_s$). A typical situation of this type is illustrated in Figure 1.

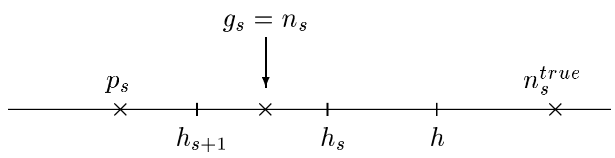

Figure 1: A typical scenario in the procedure TBS_n, where $\{1, \ldots, h\}$ is the target concept and the hypothesis $\{1, \ldots, h_s\}$ is "refuted" at step s by a false NCE g_s.

As a consequence of this clause in the definition of TBS_n one can show in Lemma 2.3 that any true NCE reduces the length of the "undetermined interval" $\{p_s, \ldots, n_s^{\text{true}} - 1\}$ by at least 50 % (as in the usual binary search procedure). The strategy of the quite different definition of the hypotheses $\{1, \ldots, h_{s+1}\}$ after a PCE at step s is to treat PCE's as a particularly valuable resource. This is justified, since PCE's are always *true* counterexamples. Therefore, instead of halving the open interval in an upwards direction, one moves the hypothesis direct up to the least unrefuted NCE, respectively the least unrefuted earlier hypothesis. The justification for this clause in the definition is given in Lemma 2.4. It is shown there that a second PCE after this move will bring definite progress: it either unmasks a false NCE's, or it definitely refutes an earlier hypothesis that was previously only "refuted" by false NCE's.

Note that the more familiar halving of the open interval in an upwards direction after each PCE (as in the regular binary search procedure) uses PCE's in a less economic fashion (if false NCE's are present). It could then occur that one uses a sequence of $\log n/2$ PCE's just to find out that one earlier NCE (to which these PCE's converge from below) was a *false* NCE.

Theorem 2.1 *Assume that f false NCE's occur in a learning process for $HEAD_n$ with learning algorithm TBS_n. Then at most $\log n$ true NCE's and at most $\log n + 3f + 1$ PCE's occur in this learning process.*

In order to prove Theorem 2.1 we analyze the properties of algorithm TBS_n in three simple Lemmata.

Lemma 2.2

a) $p_s + \left\lfloor \frac{n_s - p_s}{2} \right\rfloor \leq n_s - 1$ *for all* $s \geq 1$.

b) $p_s \leq h_{s+1} \leq n_s - 1$ *for all* $s \geq 1$.

c) If g_s is a (true or false) NCE, then $h_{s+1} \leq h_s$.

Proof:

a) Note that $p_s < n_s$ by the definition of n_s.

b) This follows from part a).

c) If $g_s \leq p_s$ then $h_{s+1} = h_s$. Assume that $g_s > p_s$ and $s > 1$. Since $h_s \leq n_{s-1} - 1$ by b), we have $n_s = g_s \leq h_s$. Furthermore $h_{s+1} = p_s + \left\lfloor \frac{n_s - p_s}{2} \right\rfloor \leq n_s$ (see part a)). ∎

Lemma 2.3 *Assume $r < s$ and g_r, g_s are true NCE's. Then* $n_s^{true} - p_s \leq \frac{n_r^{true} - p_r}{2}$.

Proof: It is obvious that $r > 1$. Thus $n_r^{true} = n_r = g_r$, since $p_r < g_r \leq h_r \leq n_{r-1} - 1$ by Lemma 2.2 b). One shows in the same way that $n_s^{true} = n_s = g_s$.

Since g_r is a NCE one has $h_{r+1} = p_r + \left\lfloor \frac{n_r^{true} - p_r}{2} \right\rfloor$.

Case 1: $p_s > h_{r+1}$
Then $p_s \geq p_r + \left\lfloor \frac{n_r^{true} - p_r}{2} \right\rfloor + 1 \geq \frac{p_r + n_r^{true}}{2}$, hence $n_s^{true} - p_s \leq n_r^{true} - p_s \leq n_r^{true} - \left(\frac{p_r + n_r^{true}}{2} \right) = \frac{n_r^{true} - p_r}{2}$.

Case 2: $p_s \leq h_{r+1}$
Then $g_j \leq h_{r+1}$ for every $j \in \{r+1, \ldots, s-1\}$ such that g_j is a PCE. Hence $h_{j+1} \leq h_{r+1}$ for each such j (by the definition of h_{j+1}). Together with Lemma 2.2 c) this implies that $h_{j+1} \leq h_{r+1}$ for every $j \in \{r+1, \ldots, s-1\}$. In particular we have shown that $h_s \leq h_{r+1}$. Thus $n_s^{true} = g_s \leq h_{r+1}$. Therefore $n_s^{true} - p_s \leq h_{r+1} - g_s \leq p_r + \left(\frac{n_r^{true} - p_r}{2} \right) - p_s \leq p_r + \left(\frac{n_r^{true} - p_r}{2} \right) - p_r = \frac{n_r^{true} - p_r}{2}$. ∎

Lemma 2.4 *Assume that g_s and g_{s+1} are PCE's. Then $g_{s+1} \geq n_s$, or $g_{s+1} > h_r$ for some $r \in \{1, \ldots, s\}$ with $p_s \leq h_r < n_s$. In other words: g_{s+1} proves that an earlier NCE was a false NCE, or it definitely refutes an earlier hypothesis $\{1, \ldots, h_r\}$ which had received a counterexample at step r, but which was at step s consistent with all currently unrefuted counterexamples.*

Proof: By construction one has $g_{s+1} > h_{s+1} = \min(\{n_s - 1\} \cup \{h_r \mid 1 \leq r \leq s \text{ and } p_s \leq h_r < n_s\})$. ∎

Proof of Theorem 2.1: Lemma 2.3 implies that at most $\log n$ true NCE's occur in any learning process with algorithm TBS_n. Hence at most $\log n + f$ NCE's occur in the considered learning process Q. Thus there exist at most $\log n + f + 1$ maximal blocks of successive PCE's in this learning process Q. Consider any such maximal block B that consists of $k + 1$ PCE's $g_s, \ldots, g_{s+k}$. Set

$$\begin{aligned} k_1^B &:= |\{j \mid \quad j \in \{s+1, \ldots, s+k\} \text{ and } g_j \geq n_{j-1}\}| \\ k_2^B &:= |\{j \mid \quad j \in \{s+1, \ldots, s+k\} \text{ and } g_j > h_r \text{ for} \\ &\qquad\qquad \text{some } r \in \{1, \ldots, j-1\} \text{ with} \\ &\qquad\qquad p_{j-1} \leq h_r < n_{j-1}\}|. \end{aligned}$$

By Lemma 2.4 we have $k_1^B + k_2^B \geq k$.

Each time when $g_j \geq n_{j-1}$ (as in the definition of k_1^B), then an earlier NCE gets proven false at step j. This happens at most once for each of the f false NCE's.

Each time when g_j provides a counterexample to an earlier hypothesis $\{1, \ldots, h_r\}$ that was consistent with all unrefuted counterexamples at the beginning of step j (as in the definition of k_2^B), then this hypothesis $\{1, \ldots, h_r\}$ can never appear to be consistent again at a later step t (since $p_t \geq g_j > h_r$). Furthermore this event can only occur if the original counterexample g_r to $\{1, \ldots, h_r\}$ was a false NCE. Thus altogether there are only f hypotheses $\{1, \ldots, h_r\}$ for which this event can ever occur.

Thus we have shown that $\sum\{k_1^B \mid B$ is a maximal block of PCE's in $Q\} \leq f$ and $\sum\{k_2^B \mid B$ is a maximal block of PCE's in $Q\} \leq f$. Altogether we have shown that at most $\log n + 3f + 1$ PCE's occur in the considered learning process Q. ∎

Remark 2.5 *One can construct in the same manner a learning algorithm TBS_n^* for the concept class*

$$TAIL_n := \{\{j, j+1, \ldots, n\} \mid \quad 1 \leq j \leq n\}$$

that satisfies an analogous version of Theorem 2.1 .

Remark 2.6 *There exist already various algorithms for binary search in the presence of two-sided errors, see e.g. [DGW], [BK]. These algorithms do not provide sufficiently strong bounds (e.g. on the number of true NCE's) to be useful for our application in section 3.*

3 A Learning Algorithm for $\mathbf{BOX_n^d}$ whose Error Bound is Polynomial in d and $\log$ n

Theorem 3.1 $LC(BOX_n^d) = O(d^2 \log n)$.

Proof: Consider any target concept $C_T = \prod_{i=1}^{d} \{a_i, \dots, b_i\} \in BOX_n^d$. The learning algorithm S for BOX_n^d issues $H_1 := \emptyset$ as its first hypothesis. If $H_1 \neq C_T$ then S receives a PCE $u = < u_1, \dots, u_d > \in C_T$. Henceforth the algorithm S splits the task of learning C_T into $2d$ separate subtasks: The learning of $\{u_i, \dots, b_i\} \subseteq \{u_i, \dots, n\}$ (i.e. of a concept from $HEAD_{n-u_i+1}$ over the transformed domain $\{u_i, \dots, n\}$) and the learning of $\{a_i, \dots, u_i\} \subseteq \{1, \dots, u_i\}$ (i.e. of a concept from $TAIL_{u_i}$) for $i = 1, \dots, d$. For each $i \in \{1, \dots, d\}$ the algorithm S employs $TBS_{n - u_i + 1}$ for the former and $TBS^*_{u_i}$ for the latter subtask.

One sets $H_2 := \{u\}$. Assume that at any step $r \geq 2$ the learning algorithm S for BOX_n^d has issued a hypothesis $H_r := \prod_{i=1}^{d} \{h_i^*, \dots, h_i\}$. Then the next hypothesis H_{r+1} is determined in the following way by the $2d$ subroutines.

Let $x = < x_1, \dots, x_d > \in C_T \Delta H_r$ be the counterexample to the hypothesis H_r of algorithm S. Note that we use the notion of a counterexample for algorithm S in the traditional sense (i.e. x is a PCE or a true NCE). If x is a PCE to hypothesis H_r, then for at least one $i \in \{1, \dots, d\}$ the point x_i is a PCE to the current hypothesis of one of the two subroutines $TBS_{n - u_i + 1}$ or $TBS^*_{u_i}$. For each such i one changes the interval in the i-th dimension according to the next hypothesis of the subroutine $TBS_{n - u_i + 1}$ resp. $TBS^*_{u_i}$. For other i one has $x_i \in \{h_i^*, \dots, h_i\}$, and one repeats in these dimensions the same interval $\{h_i^*, \dots, h_i\}$ in the next hypothesis H_{r+1} of S.

Assume now that $x = < x_1, \dots, x_d >$ is a NCE to hypothesis H_r. For each $i \in \{1, \dots, d\}$ with $x_i \neq u_i$ the point x_i provides a (true or false) NCE to the current hypothesis $\{u_i, \dots, h_i\}$ of subroutine $TBS_{n - u_i + 1}$, or to the current hypothesis $\{h_i^*, \dots, u_i\}$ of subroutine $TBS^*_{u_i}$. One updates the interval in the i-th dimension of the next hypothesis H_{r+1} of S according to the next hypothesis of $TBS_{n - u_i + 1}$ resp. $TBS^*_{u_i}$. For those i with $x_i = u_i$ one leaves the interval in the i-th dimension unchanged.

By Theorem 2.1 each subroutine for learning one of the $2d$ halfintervals encounters at most $\log n$ true NCE's. Since each NCE for algorithm S provides a true NCE for at least one of the $2d$ subroutines, S gets altogether at most $2d \log n$ NCE's. Each of these NCE's may generate false NCE's for up to $d-1$ subroutines. Hence the sum of false NCE's for all $2d$ subroutines together is $\leq (d-1)2d \log n$. Thus by Theorem 2.1 the sum of all PCE's that are received by the $2d$ subroutines is bounded by $2d(\log n + 1) + 3(d-1)2d \log n = (6d^2 - 4d)\log n + 2d$. Since each PCE to algorithm S (exept for the first one) generates a PCE for at least one of its $2d$ subroutines, the total number of PCE's that S receives is $\leq (6d^2 - 4d)\log n + 2d + 1$. Hence $LC(S) \leq 2d \log n + (6d^2 - 4d)\log n + 2d + 1 = 6d^2 \log n - 2d \log n + 2d + 1$. ∎

Remark 3.2 *Peter Auer shows in a forthcoming paper [Au] that $LC(BOX_n^d) = \Omega(d^2 \log n / \log d)$. Hence the preceding algorithm is close to optimal with regard to its error bound. He also constructs an error robust variation of our learning algorithm for BOX_n^d, that can tolerate a certain fraction of incorrect positive and negative counterexamples for the global algorithm (for any distribution of incorrect counterexamples).*

4 An Algorithm for Learning the Union of Two Boxes in the Plane

The algorithm in the preceding section was based on a solution of the credit assignment problem in which the local search procedures tolerate false negative counterexamples. It was essential for the success of this algorithm that the local search procedures never receive false positive counterexamples.

In this section we examine a more complex learning problem, in which the obvious local search procedures have to tolerate both false negative and false positive counterexamples. For any $m, n \in \mathbf{N}$ let $X_{m,n}$ be the domain

$$X_{m,n} := \{ < i, j > \mid i \in \{1, \dots, m\} \text{ and } j \in \{1, \dots, n\} \}.$$

Set $$BOX_{m,n} := \{ \{i, \dots, j\} \times \{k, \dots, l\} \mid 1 \leq i \leq j \leq m \text{ and } 1 \leq k \leq l \leq n \}.$$

We write $a := < 1, n >$ for the upper left corner and $b := < m, 1 >$ for the lower right corner of this domain

$X_{m,n}$. We consider the following concept class over the domain $X_{m,n}$:

$$TWO - BOX_{m,n} := \{ RA \cup RB \mid RA, RB \in BOX_{m,n}, \\ a \in RA \text{ and } b \in RB \}.$$

Whenever we write RA (RB) in the following, we assume that $RA \in BOX_{m,n}$ and $a \in RA$ ($RB \in BOX_{m,n}$ and $b \in RB$). Note that the two components RA and RB of a concept in $TWO - BOX_{m,n}$ may or may not intersect.

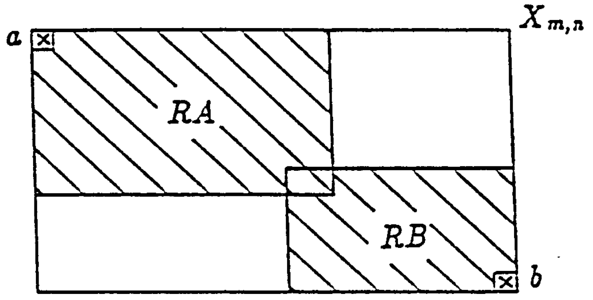

Figure 2

The learning of arbitrary target concepts $RA \cup RB$ from $TWO - BOX_{m,n}$ may be viewed as a combination of 4 search procedures that determine the lengths of the sides of RA and RB. In the same way as in the preceding section these local search procedures will receive *false negative counterexamples*, since it is not clear *which* side of RA (RB) has to be shortened in order to accomodate a NCE $g \in (RA \cup RB) - C_T$. However these local search procedures will in general also receive *false positive counterexamples*, since it is not clear whether a PCE should lie in RA, or in RB (or in both). The following result shows that nevertheless there is an efficient learning algorithm for this learning problem.

Theorem 4.1 $LC(TWO - BOX_{m,n}) = \Theta(log(m+n))$.

Proof: It is obvious that chain($TWO - BOX_{m,n}$) $= \Omega(m+n)$, where chain($\mathcal{C}$) denotes the length of the longest chain in $\mathcal{C}$ with regard to the partial order "$\leq$" of $\mathcal{C}$ defined by $C \leq C' \Leftrightarrow C \subseteq C'$. According to [MTa], [MTc] one has

$$LC(TWO - BOX_{m,n}) \geq \\ \lfloor \log(\text{chain}(TWO - BOX_{m,n})) \rfloor.$$

In fact, the same lower bound holds for $LC - ARB(TWO - BOX_{m,n})$.

In order to prove the upper bound of Theorem 4.1, we first consider the following subclass of $TWO - BOX_{m,n}$:

$$\mathcal{U}_{m,n} := \{ RA \cup RB \mid \; RA, RB \in BOX_{m,n}, \\ a \in RA, b \in RB, \text{ and } |RA \cap RB| = 1 \}.$$

We will exhibit in the proof of the main lemma (Lemma 4.3) an efficient learning algorithm K for this concept class $\mathcal{U}_{m,n}$. This algorithm K will employ as local search procedures the following binary search algorithm CBS ("conservative binary search"), which is distinguished by the property that it never receives two successive NCE's. Although CBS will also be used in a nonstandard situation (where there exists no target concept), it suffices that we analyze it here in the context of the basic learning model that was defined in section 1.

Definition of the binary search algorithm CBS for learning $HEAD_n$:

Assume that the environment has fixed some target concept $C_T \in HEAD_n$. At step r the learner issues the hypothesis $H_r := \{1, \ldots, r\} \in HEAD_n$. If $H_r \neq C_T$, he receives at step r a counterexample $g_r \in H_r \Delta C_T$. Let p_s be the maximum of 1 and the largest PCE received by the end of step s, and let n_s be the minimum of $n+1$ and the smallest NCE received by the end of step s.

Set $h_1 := 1$, and

$$h_{s+1} := \begin{cases} p_s & \text{, if } g_s \text{ is a NCE} \\ p_s + \left\lfloor \frac{n_s - p_s}{2} \right\rfloor & \text{, if } g_s \text{ is a PCE.} \end{cases}$$

Lemma 4.2 *Consider any learning process for learning $HEAD_n$ (in the basic learning model) with learning algorithm CBS. Then there are never two successive NCE's in this learning process. Furthermore $n_t - p_t \leq \frac{n}{2^{\lfloor t/2 \rfloor}}$ for any step $t \geq 1$ with $H_t \neq C_T$.*

Proof: The first claim is obvious from the construction of CBS.

In order to prove the second claim we show that for any $s \geq 1$ such that g_s is a PCE and $H_{s+1} \neq C_T$ one has

$$n_{s+1} - p_{s+1} \leq \frac{n_s - p_s}{2}.$$

Since g_s is a PCE we have

$$h_{s+1} = p_s + \left\lfloor \frac{n_s - p_s}{2} \right\rfloor.$$

If g_{s+1} is a NCE we have

$$n_{s+1} \leq h_{s+1} \leq p_s + \frac{n_s - p_s}{2}$$

and $p_{s+1} = p_s$, hence

$$n_{s+1} - p_{s+1} \leq \frac{n_s - p_s}{2}.$$

If g_{s+1} is a PCE we have

$$p_{s+1} \geq 1 + h_{s+1} \geq p_s + \frac{n_s - p_s}{2}$$

and $n_{s+1} = n_s$, hence

$$n_{s+1} - p_{s+1} \leq n_s - \left(p_s + \frac{n_s - p_s}{2}\right) = \frac{n_s - p_s}{2}.$$

∎

Lemma 4.3 *(Main Lemma)*
$LC^{TWO-BOX_{m,n}}(\mathcal{U}_{m,n}) = O(\log(m+n))$.

Remark 4.4

a) *One has to use in Lemma 4.3 a larger hypothesis space than* $\mathcal{U}_{m,n}$, *since* $LC(\mathcal{U}_{m,n}) = \Omega(m+n)$. *This lower bound can be shown with the help of an adversary strategy that gives only negative counterexamples from the "diagonal line" between* $< 1,1 >$ *and* $< m,n >$ *in* $X_{m,n}$.

b) *The learning algorithm for* $\mathcal{U}_{m,n}$ *that is constructed in the proof of Lemma 4.3 uses actually only the hypothesis space* $\{H \in TWO-BOX_{m,n} \mid \overline{H} \in TWO-BOX_{n,m}\}$, *which is a proper subclass of* $TWO-BOX_{m,n}$.

Proof of Lemma 4.3: In order to design an efficient learning algorithm K for $\mathcal{U}_{m,n}$, we note that any concept $RA \cup RB \in \mathcal{U}_{m,n}$ with $a \in RA, b \in RB$ and $|RA \cap RB| = 1$ can be uniquely characterized by the single intersection point $w =< i,j >$ of the rectangles RA and RB. We write R_w for this concept $RA \cup RB$ from $\mathcal{U}_{m,n}$, and RA_w, RB_w for its two components RA, RB.

The learning algorithm K for $\mathcal{U}_{m,n}$ proceeds in a recursive manner. Assume that it has already exhibited an $m' \times n'$ rectangle $W \subseteq X_{m,n}$ with $w \in W$ for the target concept $R_w \in \mathcal{U}_{m,n}$ (initially one has $W = X_{m,n}$). We will use area(W) (or rather: 1-area(W)) as a "measure of progress" for the learning algorithm K. We will not be able to guarantee that area(W) can always be reduced by a fixed fraction within $O(1)$ steps of learning algorithm K. However we can show that there is some number t (which depends on the specific learning process) such that K produces in $t+2$ further steps a rectangle $\tilde{W} \subsetneqq W$ with $w \in \tilde{W}$ and

$$\text{area}(\tilde{W}) \leq \frac{\text{area}(W)}{\max(2, 2^{t/4-2})}.$$

Assume that $e \in X_{m,n}$ is the "centerpoint" of W. We first consider the case where e does not lie on the perimeter of W (i. e. we assume that $m' > 2$ and $n' > 2$). Then K issues R_e as its next hypothesis.

We will analyze separately the two cases where the learner receives a positive respectively negative counterexample g to this hypothesis R_e. In each case the primary goal of the learner is to determine whether $w \in S$ or $w \in T$, where S and T are the two rectangles that are defined by counterexample g as indicated in Figure 3. However the learner may not be able to achieve this information within a fixed number of steps. Instead, he enters a "P-phase" (in case 1), respectively an "N-phase" (in case 2). We are not able to bound the number t of steps which are spent by the learner in the respective phase. However we can guarantee that at the end of such phase the learner has not only determined whether $w \in S$ or $w \in T$, but in addition he can exhibit a rectangle $\tilde{W} \subsetneqq W$ with

$$\text{area}(\tilde{W}) \leq \frac{\text{area}(W)}{\max(2, 2^{t/4-2})}$$

Obviously this suffices in order to determine the target concept from $\mathcal{U}_{m,n}$ in altogether $O(\log(m+n))$ learning steps.

Case 1: The learner receives a positive counterexample $g =< p_g, q_g > \in R_w - R_e$

We can assume without loss of generality that $g \in W$. If g lies to the left (right) of W, we may replace it by the point on the same row in the leftmost (rightmost) column of W. If g lies above (below) W, we may replace it by a point in the same column in the highest (lowest) row of W. Note that we have used here the assumption that e does not lie on the perimeter of W (i. e. $m' > 2$ and $n' > 2$). Furthermore it is clear that g cannot lie in the area to the left and above W (respectively to the right and below W), since these two areas are contained both in the target concept R_w (because $w \in W$) and in the hypothesis R_e.

Let $S, T \subseteq W$ be the rectangles with $S \cap T = \{g\}$ and $R_g \cap W = S \cup T$, as shown in Figure 3. It is clear that $w \in S \cup T$. The algorithm K issues R_g as its next hypothesis. If $R_g \neq C_T$ it follows that $w \in S \cup T - (S \cap T)$. In order to determine whether $w \in S$ or $w \in T$, the learning algorithm K enters a procedure that we call a *P-phase*. When this P-phase terminates after t steps, the algorithm exhibits a rectangle $\tilde{W}$ with $\tilde{W} \subseteq S$ or $\tilde{W} \subseteq T$ such that $w \in \tilde{W}$ and $\text{area}(\tilde{W}) \leq \frac{\text{area}(W)}{2^{t/4-2}}$.

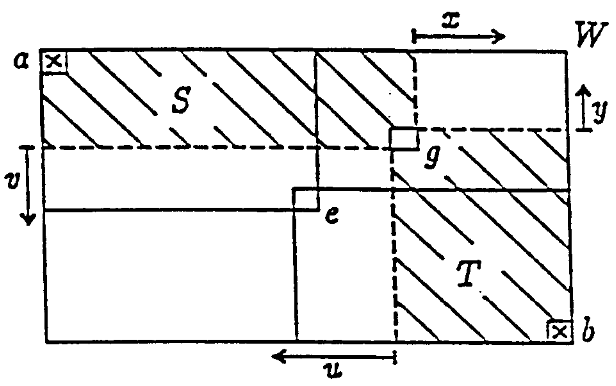

Figure 3

The P-phase consists of 4 concurrent binary search procedures that try to determine the values of 4 parameters x, y, u, v (see Figure 3). If $w \in T$, then the values of x and v give the horizontal resp. vertical distance of w from g, whereas the parameters y and u are undefined. If $w \in S$, then the values of u and y give the horizontal resp. vertical distance of w from g, whereas the parameters x and v are undefined. The hypothesis of algorithm K is at each step of the P-phase of the form $RA \cup RB$ with $a \in RA, S \subseteq RA, b \in RB$, and $T \subseteq RB$. The exact lengths of the sides of RA (RB) are determined by the current hypotheses of the binary search procedures for x and v (u and y). The remainder of our analysis of case 1 is devoted to the precise description and analysis of the P-phase.

The difficulty of the P-phase is caused by the need to carry out the concurrent binary search procedures for the parameters x, y, u, v without knowing whether $w \in S$ or $w \in T$, and hence without knowing which ones of x, y, u, v are actually undefined. Thus we have to combine two "real" binary search procedures with two "dummy" binary search procedures, without knowing which are the real ones. The danger is that we may spend many learning steps exclusively for the benefit of those search procedures that later turn out to be "dummy" (i. e. they search for the values of parameters that are actually undefined). Consider for example the two search procedures for the parameters x and y. We know that exactly one of those two parameters is undefined. If one receives a PCE $q \in R_w - (RA \cup RB)$ in the region above T (see Figure 3), then this provides a PCE for both of the two binary search procedures for x and for y (in particular also for the "real" one among the two). However a NCE $q \in (RA \cup RB) - R_w$ in the region above T may provide a NCE only for one of these two binary search procedures. If one has bad luck, it provides a NCE only for the one that later turns out to be "dummy", and no progress has been made at this learning step for the "real" binary search procedure among the two.

This difficulty is handled by using for the local binary search procedures the algorithm CBS that was analyzed in Lemma 4.2. It may still occur then, that a NCE provides progress only for the "dummy" one among two binary search procedures CBS. However since *no* binary search procedure CBS (even the "dummy" ones) may receive two NCE's in a row, this event can occur on average at most at every second step.

An exception may occur at a step where a binary search procedure that searches for an undefined parameter receives a NCE $g_s \leq p_s$ (and hence possibly two NCE's in a row). However such step (which reveals to the learner *which* ones of the parameters are undefined) automatically terminates the current P-phase.

We now describe in detail how the algorithm K proceeds during the considered P-phase. One should keep in mind that this P-phase focuses its activity on the $m' \times n'$-rectangle $W \subseteq X_{m,n}$, but that its hypotheses are required to be from $TWO - BOX_{m,n}$. One carries out 4 concurrent binary searches with algorithm CBS. The first one of these is a copy of CBS that searches for the value of parameter x, in case that x is defined. More precisely: CBS searches for the concept $\{0, 1, \ldots, x\} \in HEAD_{\hat{n}}$ for some $\hat{n} \leq n$ (for technical reasons we take here $\{0, \ldots, \hat{n} - 1\}$ as domain for $HEAD_{\hat{n}}$, instead of $\{1, \ldots, \hat{n}\}$). The second binary search procedure is a copy of CBS that searches for the value of y, in case that y is defined. Analogously one uses copies of CBS to search for u resp. v. Assume that so far none of these 4 copies of CBS has encountered a contradiction among its counterexamples, and that h_x, h_y, h_u, h_v are the endpoints of the current hypotheses $\{0, \ldots, h_x\}, \{0, \ldots, h_y\}, \{0, \ldots, h_u\}, \{0, \ldots, h_v\}$ in the respective copies of CBS. Then the algorithm K issues as its next hypothesis the following concept $H \in TWO - BOX_{m,n}$:

$$\begin{aligned} H \;:=\; & \{1, \ldots, p_g + h_x\} \times \{q_g - h_v, \ldots, n\} \\ & \cup\ \{p_g - h_u, \ldots, m\} \times \{1, \ldots, q_g + h_y\}. \end{aligned}$$

It is obvious that $S \cup T \subseteq H$.

Let $h \in H \,\Delta\, R_w$ be a counterexample to this hypothesis. We will first consider the case where $h \in W$. We will analyze in the next two paragraphs the subcase where $h \in W$ is a PCE. In the subsequent third paragraph we will analyze the subcase where $h \in W$ is a NCE. After that we will turn to the analysis of the case where $h \notin W$.

If h is a PCE and if h lies above T, then one processes the two coordinates of h as PCE's for the two copies of CBS that search for the parameters x and y.

If $h \in W$ is a PCE that lies to the left of T, one processes the two coordinates of h as PCE's for the two copies of CBS that search for the parameters u and v.

If $h \in W$ is a NCE with $h \notin S \cup T$ and h lies above T, then the two coordinates of h provide a NCE for *at least one* of the two copies of CBS that search for x and y. (Since h cannot be guaranteed to provide a NCE for both copies of CBS, it may potentially only provide a NCE for the "dummy" copy of CBS). If $h \in W$ is a NCE with $h \notin S \cup T$ and h lies to the left of T, then the two coordinates of h provide a NCE for *at least one* of the two copies of CBS that search for u and v.

If $h \in W$ is a NCE with $h \in S \cup T$ then it terminates the current P-phase. If $h \in T$, then it is proven that $w \in T$, and that the parameters y and u are undefined. The current P-phase also ends if in any of the preceding cases at least one of the 4 copies of CBS receives a CE that contradicts another CE that it had received at an earlier step. Assume for example that the copy of CBS that searches for x receives a PCE (NCE) that contradicts an earlier NCE (PCE). This implies that the parameter x is undefined. Hence one has $w \in S$, which implies that the parameter v is also undefined.

Finally we consider the case where the counterexample $h \in H \Delta R_w$ does not lie in W. If h is a PCE to the right of W, then it implies that $w \in S$. Hence this counterexample terminates the current P-phase. If h is a NCE to the right of W, it provides a NCE for the binary search for y (but no CE for the binary search for x). The cases where h lies above, below, or left of W are handled analogously.

It remains to be shown that in each possible case where the current P-phase is terminated, one can not only decide whether $w \in S$ or $w \in T$, but one can also exhibit an axis-parallel rectangle $\tilde{W}$ with $\tilde{W} \subseteq S$ or $\tilde{W} \subseteq T$, $w \in \tilde{W}$, and $\text{area}(\tilde{W}) \leq \frac{\text{area}(W)}{\max(2, 2^{t/4-2})}$, where t is the number of counterexamples that have been received during the current P-phase.

Each PCE h that is received before the end of the P-phase provides a PCE for both of the binary search procedures for x and y, or for both of the binary search procedures for u and v. Each NCE provides a NCE for at least one of the binary search procedures for x and y, or for at least one of the binary search procedures for u and v. Since none of these 4 copies of procedure CBS (not even those that search for undefined parameters) can receive two successive NCE's (except at the last step of this P-phase), at least $t' := \frac{t}{2} - 3$ of the t counterexamples of this P-phase provide CE's for one of the two copies of CBS which search for parameters that are actually defined. By Lemma 4.2 at least $\lfloor \frac{t'}{2} \rfloor$ of them reduce one of the two dimensions of $\tilde{W}$ by at least 50 %, starting with S resp. T. Since area(S), area(T)$\leq \frac{\text{area}(W)}{2}$, one has

$$\text{area}(\tilde{W}) \leq \frac{\text{area}(W)}{2^{1+\lfloor \frac{1}{2}(t/2-3) \rfloor}} \leq \frac{\text{area}(W)}{2^{t/4-2}}.$$

Finally we observe that any P-phase terminates at the latest after $O(\log(m+n))$ step, since each single one of the 4 procedures CBS can receive at most $2\log(m+n)$ counterexamples without running into a contradiction.

Case 2: The learner receives a negative counterexample $g = < p_g, q_g > \in R_e - R_w$.

If $g \notin W$ then we can immediately exhibit a rectangle $\tilde{W} \subseteq W$ with $w \in \tilde{W}$ and $\text{area}(\tilde{W}) \leq \frac{\text{area}(W)}{2}$. Since $g \in R_e = RA \cup RB$, we know that g lies either in the rectangle RA of R_e that contains the upper left corner a of the domain $X_{m,n}$, or g lies in the rectangle RB that contains the lower right corner b of the domain $X_{m,n}$. If $g \in RA$ and g lies above W, then $g \notin R_w$ implies that w lies in the left half of W. If $g \in RA$ and g lies left of W, then $g \notin R_w$ implies that w lies in the upper half of W. Hence in either case we can exhibit a rectangle $\tilde{W}$ with $w \in \tilde{W}$ and $\text{area}(\tilde{W}) \subseteq \frac{\text{area}(W)}{2}$. The argument for $g \in RB$ is analogous.

We now assume that $g \in W$.

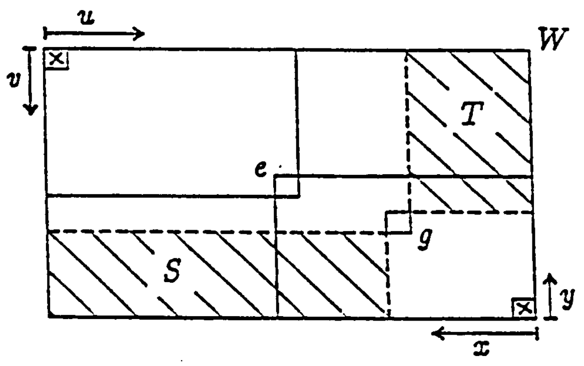

Figure 4

It is then clear that $w \in S \cup T$ for the rectangles S, T that are defined by g as indicated in Figure 3. In order to determine whether $w \in S$ or $w \in T$ the algorithm then enters an *N-phase*. An N-phase consists of 4 concurrent binary searches that determine the values of 4 parameters x, y, u, v. If $w \in T$, then the value of x is the horizontal distance of w from the rightmost column of W, the value of v is the vertical distance of w from the top row of W, and the parameters y and u are undefined.

If $w \in S$, then the value of y is the vertical distance of w from the bottom row of W, u is the horizontal distance of w from the leftmost column of W, and the parameters x and v are undefined. Each hypothesis during the N-phase is the union of two rectangles RA and RB. RA is contained in $\{1, \ldots, p_g - 1\} \times \{q_g + 1, \ldots, n\}$, and the lengths of its sides are determined by the binary search procedures for u and v. Analogously RB is contained in $\{p_g + 1, \ldots, m\} \times \{1, \ldots, q_g - 1\}$, and the lengths of its sides are determined by the binary search procedures for x and y. In order to verify Remark 4.4 b) we note that $\overline{RA \cup RB} \in TWO - BOX_{n,m}$ since no row and no column contains points from both RA and RB.

In contrast to the situation in a P-phase, a PCE h to hypothesis $RA \cup RB$ may yield a PCE *only for one* of the search procedures for u and v (if h lies above S), or *only for one* of the search procedures for x and y (if h lies below T). On the other hand, a NCE to hypothesis $RA \cup RB$ provides a NCE either for *both* search procedures for u and v , or for *both* search procedures for x and y. Hence one uses here as binary search procedures for x, y, u, v a dual version CBS' of CBS, for which no two successive PCE's can occur.

Note that any PCE $h \in \{1, \ldots, p_g\} \times \{1, \ldots, q_g\} \cup \{p_g, \ldots, m\} \times \{q_g, \ldots, n\}$ reveals whether $w \in S$ or $w \in T$, and it will automatically terminate this N-phase. Similarly a NCE outside of W decides whether $w \in S$ or $w \in T$, and it also terminates this N-phase. Apart from these cases, the N-phase is also terminated by any counterexample that provides a contradiction to an earlier counterexample for any of the 4 copies of the binary search procedure CBS'. The rest of the analysis of the N-phase is analogous to that of the P-phase.

Finally we have to comment on the case where the algorithm K cannot continue its recursion with a P-phase or an N-phase, because it has already narrowed down the location of w to an $m' \times n'$ rectangle $W \subseteq X_{m,n}$ with $m' \leq 2$ or $n' \leq 2$. Assume for example that $m' = 2$. Then the algorithm K carries out for both columns in W a straightforward binary search for w. This is possible, because for the binary search in the "correct" column it can interprete each counterexample without ambiguity.

This completes the proof of Lemma 4.3. ∎

With the help of the preceding main lemma we are now able to prove Theorem 4.1. Besides the cornerpoints $a =< 1, n >$ and $b =< m, 1 >$ we will also distinguish the other two cornerpoints $c =< 1, 1 >$ and $d =< m, n >$ of the domain $X_{m,n}$. For many $C \in TWO - BOX_{m,n}$ the complement $\overline{C} := X_{m,n} - C$ may be viewed as element of $BOX_{m,n}, BOX_{n,m}$ or $TWO - BOX_{n,m}$. In order to consider $\overline{C}$ as element of $TWO - BOX_{n,m}$ one "turns the domain by 90°", i. e. one identifies $X_{m,n}$ with $X_{n,m}$, a with c, and b with d (see Figure 4). This duality is frequently exploited in the following in order to discuss subroutines that aim at learning $\overline{C}_T$ instead of C_T. This makes sense for those cases where $\overline{C}_T$ has a simpler structure than C_T. Note however that one has to be careful when one exploits this duality, since there are $C \in TWO - BOX_{m,n}$ for which $\overline{C}$ can not be interpreted as a union of one or two rectangles (e. g. consider $C = \{1, \ldots, m-1\} \times \{n\} \cup \{2, \ldots, m\} \times \{1\}$).

The learning algorithm L for $TWO - BOX_{m,n}$ proceeds in 4 phases. The first hypothesis of the first phase is the set $X_{m,n}$. If $X_{m,n} \neq C_T$, then one receives a NCE. It is then clear that $c \notin C_T$ or $d \notin C_T$. In order to eliminate the case where $\{c, d\} \cap C_T \neq \emptyset$, one uses as a subroutine some learning algorithm A for $BOX_{m,n}$ that is guaranteed to find any $C \in BOX_{m,n}$ in $O(\log(m+n))$ steps, using hypotheses from $BOX_{m,n}$ (see section 3, or [MTb]). One first executes this learning algorithm A in order to find $\overline{C}_T$ under the assumption that $c \in \overline{C}_T$ and $d \in C_T$ (hence $\overline{C}_T \in BOX_{m,n}$) by inverting the "sign" of each example and by replacing each hypothesis H of A by its complement $\overline{H}$ (note that $\overline{H} \in TWO - BOX_{m,n}$ for any $H \in BOX_{m,n}$ with $c \in H$). In this way one finds C_T in $O(\log(m+n))$ steps if $c \in \overline{C}_T$ and $d \in C_T$. If this attempt is not successful, one executes A again in order to find $\overline{C}_T$, but this time under the assumption that $d \in \overline{C}_T$ and $c \in C_T$. If this attempt is also not successful, one has proven that $c \in \overline{C}_T$ and $d \in \overline{C}_T$.

During its second phase the algorithm L checks whether $\overline{C}_T \in \mathcal{U}_{n,m}$. For this purpose it executes the algorithm K form the proof of Lemma 4.3 for $O(\log(m+n))$ steps over the domain $X_{n,m}$ in order to learn the complement of C_T. Hence the sign of each example is inverted, and each hypothesis H of algorithm K is replaced by its complement $\overline{H}$. Note that according to Remark 4.4 b) this algorithm K for $\mathcal{U}_{n,m}$ uses only hypotheses H such that $\overline{H} \in TWO - BOX_{m,n}$. Hence $\overline{H}$ is a permissible hypothesis for algorithm L. If this simulation of K fails to identify $\overline{C}_T$ within its alloted time, we know that $\overline{C}_T \notin \mathcal{U}_{n,m}$. Furthermore the sample S that has been assembled by this time has the property that no $C \in TWO - BOX_{m,n}$ with $\overline{C} \in \mathcal{U}_{n,m}$ is consistent with S. (We refer to a set of positive and/or negative examples for C_T as a *sample* for C_T.)

During its third phase the learning algorithm L checks whether C_T is of the form $RA \cup RB$ with $RA \cap RB \neq \emptyset$. One uses here the following simple structural result.

Lemma 4.5 *Assume that S is a sample that is* *consistent with some $C = RA \cup RB \in TWO-BOX_{m,n}$ such that $c, d \notin C$ and $RA \cap RB \neq \emptyset$. Furthermore assume that S is not consistent with any $C \in TWO-BOX_{m,n}$ such that $\overline{C} \in \mathcal{U}_{n,m}$.*

Then there do not exist among those concepts that are consistent with S for $i = 1, 2$ concepts $C_i = RA_i \cup RB_i$ with $RA_i \cap RB_i \neq \emptyset$ and $\overline{C}_i$ of the form $RC_i \cup RD_i$ with $RC_i, RD_i \in BOX_{m,n}, c \in RC_i, d \in RD_i$, such that $g \in RC_1 \cap RD_2$ for some negative example g in S.

In other words: the assignment of negative examples in S to the two rectangles of $\overline{C}$ is unique (for concepts $C = RA \cup RB$ with $RA \cap RB \neq \emptyset, c, d \notin C$).

Proof of Lemma 4.5: Assume for a contradiction that there exist such concepts C_1, C_2 and such negative example $g \in RC_1 \cup RD_2$ in S. Then there are two different rows (resp. columns) r_1, r_2 in the domain $X_{m,n}$ such that $r_1 \subseteq C_1, r_2 \subseteq C_2$, and g lies strictly between r_1 and r_2. Without loss of generality we assume that r_1, r_2 are rows with r_1 above r_2 (see Figure 5).

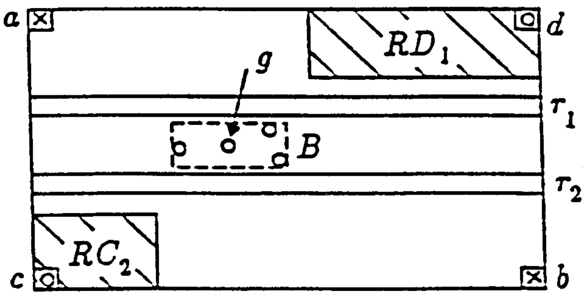

Figure 5

Let B be the smallest axis parallel rectangle that contains all negative examples in S which lie between r_1 and r_2. Note that $g \in B$. Since C_1 is consistent with S, all negative examples in S that lie above r_1 are contained in RD_1, hence they lie to the right of the rightmost column of B. Since C_2 is consistent with S, all negative examples in S that lie below r_2 are contained in RC_2, hence they lie to the left of the leftmost column of B. In particular all negative examples in S are contained in $B \cup RC_1 \cup RD_2$.

Since $B \subseteq RC_1 \cap RD_2$ we can define rectangles $\tilde{RC} \subseteq RC_1$ and $\tilde{RD} \subseteq RD_2$ with $c \in \tilde{RC}$ and $d \in \tilde{RD}$ such that $\tilde{RC}$ and $\tilde{RD}$ intersect exactly at the top right corner of B. Hence $\tilde{RC} \cup \tilde{RD} \in \mathcal{U}_{n,m}$. By construction we have $B \cup RC_1 \cup RD_2 \subseteq \tilde{RC} \cup \tilde{RD}$, hence $\tilde{RC} \cup \tilde{RD}$ contains all negative examples in S. Furthermore $\tilde{RC} \cup \tilde{RD}$ does not contain any positive examples in S, since $RC_1 \cup RD_2$ does not contain any positive example in S and $\tilde{RC} \cup \tilde{RD} \subseteq RC_1 \cup RD_2$. Hence the complement of $\tilde{RC} \cup \tilde{RD}$ is a concept $C \in TWO-BOX_{m,n}$ that is consistent with S and which satisfies $\overline{C} = \tilde{RC} \cup \tilde{RD} \in \mathcal{U}_{n,m}$. However such concept C does not exist by the assumption of Lemma 4.5. ∎

Remark 4.6 *We would like to point out that the unique partition of negative examples in S (that exists by Lemma 4.5) can be computed in an efficient manner. One can assume without loss of generality that RC and RD are "spanned" by c (resp. d) and the negative examples in S that are assigned to them. Hence it suffices to cycle through all pairs p_1, p_2 of negative examples in S and check whether the rectangle that is spanned by $\{c, p_1, p_2\}$ is a feasible solution for RC.*

The strategy of L during its third phase is the following. It employs 4 concurrent copies of the dual version TBS' of the error tolerant binary search procedure TBS from section 2. TBS' "tolerates" false PCE's (in a sense analogous to Theorem 2.1), as long as it receives only true NCE's. If $C_T = RA \cup RB$ with $RA \cap RB \neq \emptyset$ then these 4 copies of TBS' will find the lengths x, y, u, v of the sides of the rectangles RA, RB (see Figure 6).

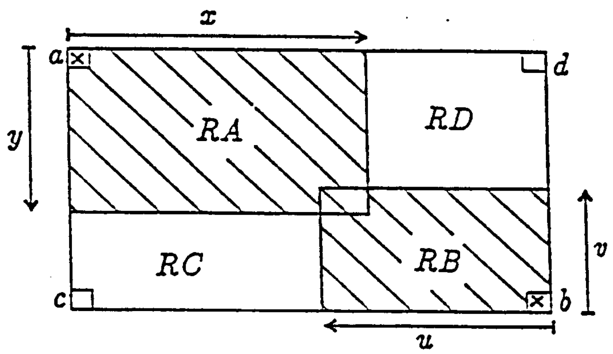

Figure 6

Let $\tilde{S}$ be any extension of the so far collected set S of examples by further examples for C_T. Then $\tilde{S}$ satisfies the assumptions of Lemma 4.5 (provided that C_T is of the form $RA \cup RB$ with $RA \cap RB \neq \emptyset$). Hence one can uniquely (and efficiently) assign any negative example in $\tilde{S}$ to one of the rectangles $RC, RD \in BOX_{m,n}$ with $RC \cup RD = \overline{C}_T, c \in RC,\ d \in RD$. Obviously any negative example that has been assigned to RC (RD) provides true negative examples for the binary search procedures for u and y (x and v). This is the reason why false NCE's can be avoided in the 4 concurrent binary searches of this phase. The hypothesis H of L

during this phase will always be of the form

$$H := \{1,\ldots,h_x\} \times \{n-h_y+1,\ldots,n\} \cup \{m-h_u+1,\ldots,m\} \times \{1,\ldots,h_v\},$$

where h_x, h_y, h_u, h_v are the endpoints of the hypotheses for the associated binary search procedures TBS'. Whenever one receives a NCE $g \in H - C_T$, one determines the unique assignment of g to RC or RD (among all $C \in TWO-BOX_{m,n}$ that are consistent with all examples received so far, and which satisfy $C_T = RA \cup RB$ with $RA \cap RB \neq \emptyset$). Hence the coordinates of g provide true NCE's for one or two copies of TBS' (and no false NCE for any copy of TBS'). On the other hand the coordinates of a PCE $g \in C_T - H$ are interpreted as positive examples for all copies of TBS'. Hence g provides a true PCE for at least one copy of TBS', and false PCE's for up to 3 copies of TBS'.

An analogous version of Theorem 2.1 for TBS' implies that all 4 copies of TBS' together can receive at most $4\log(m+n)$ true PCE's. Hence at most $12\log(m+n)$ false PCE's, and consequently at most $1+37\log(m+n)$ NCE'S can be received altogether by the 4 copies of TBS' that are employed by L during this third phase. If in fact $C_T = RA \cup RB$ with $RA \cap RB \neq \emptyset$, then L will identify C_T during this phase. Of course we terminate this phase if it has not lead to the identification of C_T within its alloted time, or if it runs into some contradiction (which can only arise if C_T is not of the conjectured form).

If the third phase of L has not suceeded in identifying C_T, one may conclude that the set S of examples that has been collected up to this point is not consistent with any $C = RA \cup RB \in TWO-BOX_{m,n}$ such that $RA \cap RB \neq \emptyset$. Hence we can apply the following simple structural result.

Lemma 4.7 *Assume that S is a sample that is consistent with some $C \in TWO-BOX_{m,n}$. Furthermore assume that S is not consistent with any $C = RA \cup RB \in TWO-BOX_{m,n}$ with $RA \cap RB \neq \emptyset$.*

Then there do not exist two concepts $C_1, C_2 \in TWO-BOX_{m,n}$ that are both consistent with S, and where $C_i = RA_i \cup RB_i, a \in RA_i, b \in RB_i$, and the rectangles RA_i and RB_i are separated by horizontal lines for $i = 1,2$, such that $g \in RA_2 \cap RB_1$ for some positive example g in S.

In other words: the assignment of positive examples in S to RA, RB is unique among all consistent concepts $C = RA \cup RB$ whose components RA, RB are separated by horizontal lines.

The analogous result holds for concepts whose components are separated by vertical lines.

Proof of Lemma 4.7: Assume for a contradiction that there exist such concepts $C_1 = RA_1 \cup RB_1$ and $C_2 = RA_2 \cup RB_2$ (whose components are separated by two horizontal lines) and some positive example g in S with $g \in RA_2 \cap RB_1$. Then $RA_2 \cup RB_1$ is a concept in $TWO-BOX_{m,n}$ that is consistent with S, and whose components RA_2, RB_1 have a nonempty intersection. However such concept does not exist by the assumption of Lemma 4.7. ∎

During its fourth phase the algorithm L first checks whether $C_T = RA \cup RB$ for rectangles RA, RB that are separated by some horizontal line. In the same way as in phase 3 it employs 4 concurrent binary search procedures that search for the lengths x, y, u, v of the sides of RA, RB (see Figure 4). Each hypothesis H of L is constructed from the current hypotheses of the 4 binary search procedures in the same way as in phase 3. However during this phase we use for these procedures instead of TBS' the original error tolerant binary search procedure TBS from section 2 (which "tolerates" false NCE's but no false PCE's).

Whenever a PCE $g \in C_T - H$ is given to L, it can decide with the help of Lemma 4.7 whether $g \in RA$ or $g \in RB$ (provided that $C_T = RA \cup RB$ for rectangles RA, RB that are separated by a horizontal line). Hence it can give the coordinates of g as true positive examples to those copies of TBS that search for x and y (if $g \in RA$), resp. to those copies of TBS that search for u and v (if $g \in RB$). Since $g \notin H$, it will provide a true PCE for at least one of these 4 copies of TBS (but no false PCE for any of them).

Any NCE $g \in H - C_T$ for the hypothesis H of L provides a true NCE for at least one of the 4 copies of TBS (and false NCE's for up to three copies of TBS).

If C_T consists in fact of two rectangles that are separated by some horizontal line, L will identify C_T during this phase in at most $1+37\log(m+n)$ steps (by Theorem 2.1).

If L does not identify C_T in this way, we know that the components RA, RB of C_T are separated by a vertical line. Hence it suffices to repeat the preceding process for the case of vertical separations.

Each phase of L takes at most $O(\log(m+n))$ steps. Hence the proof of Theorem 4.1 is now complete. ∎

Remark 4.8 *With regard to the general structure of the proof of Theorem 4.1 we would like to point out that it is necessary to apply the main lemma (Lemma 4.3) to the complements of the concepts* $C \in TWO - BOX_{m,n}$, *rather than to the concepts themselves. This arises from a rather subtle aspect of the third phase of the algorithm. This third phase relies on the structural result of Lemma 4.5, which does not have an appropriate "dual version" (with* C *and* $\overline{C}$ *interchanged). A source of this asymmetry is the fact that the two rectangles* RC, RD *which form the complement of some* $C = RA \cup RB \in TWO - BOX_{m,n}$ *with* $RA \cap RB \neq \emptyset$ *have no common row or column. However the two components* RA, RB *of some* $C = RA \cup RB \in TWO - BOX_{m,n}$ *with* $RA \cap RB = \emptyset$ *may very well have a common row or column.*

Remark 4.9 *One can use the algorithm* L *from Theorem 4.1 as a subroutine in order to get an efficient learning algorithm for the concept class* $\mathcal{U}-2-BOX_n^2 := \{C_1 \cup C_2 \mid C_1, C_2 \in BOX_n^2\}$. *One starts each learning process by executing a learning algorithm for* BOX_n^2, *until one has collected a sample* S *that is not consistent with any* $C \in BOX_n^2$. *It is easy to show that any such sample* S *contains two positive examples* a, b *and a negative example* q *s.t.* q *lies in the rectangle* R *that is spanned by* a *and* b. *It is then clear that* a *and* b *lie in different components of* $C_T \in \mathcal{U} - 2 - BOX_n^2$. *This implies that* $C_T \cap R \in TWO - BOX_{\tilde{m},\tilde{n}}$ *for* $X_{\tilde{m},\tilde{n}} := R$. *Hence one can apply the learning algorithm* L *from Theorem 4.1 over the domain* R *in order to learn* $C_T \cap R$, *and separate learning algorithms for* BOX *(resp.* $TWO - BOX$*) for other parts of the domain. At each step the hypothesis of the resulting learning algorithm for* $\mathcal{U} - 2 - BOX_n^2$ *is the union of the hypotheses that result from the subroutines for various parts of the domain. One gets in this way a learning algorithm that is guaranteed to find* C_T *in* $O(\log n)$ *step, but whose hypotheses consist of more than 2 rectangles.*

5 Open Problems

One challenging open problem is posed by the gap between our upper bound $O(d^2 \log n)$ and Auer's [Au] lower bound of $\Omega(d^2 \log n / \log d)$ for $LC(BOX_n^d)$.

Furthermore most questions concerning the on-line learning complexity of the concept class

$$\mathcal{U} - k - BOX_n^d := \{C_1 \cup \ldots \cup C_k | C_1, \ldots, C_k \in BOX_n^d\}$$

are still open. In particular, it is open whether $LC(\mathcal{U} - k - BOX_n^d) = O(\text{poly}(\log n, d))$ for constant $k \geq 2$.Even for the special case $k = d = 2$ it is not known whether $LC(\mathcal{U} - 2 - BOX_n^2) = O(\log n)$ (although there is a positive result with a slightly larger hypothesis space; see Remark 4.9). Very recently Chen [C] has shown that $LC(\mathcal{U} - 2 - BOX_n^2) = O(\log^2 n)$.

6 Acknowledgement

We would like to thank David Haussler for drawing our attention to the problem of efficient on-line learning of rectangles in arbitrary dimensions. Furthermore we are grateful to two anonymous referees for their helpful comments.

References

[A] D. Angluin, "Queries and concept learning", *Machine Learning*, 2, 1988, 319 - 342.

[Au] P. Auer, "On-line learning of rectangles in noisy environments", *Proc. of the 6th Annual ACM Conference on Computational Learning Theory*, 1993, 253 - 261.

[BK] R. S. Borgstrom, S. R. Kosaraju, "Comparison-based search in the presence of errors", *Proc. of the 25th Annual ACM Symposium on the Theory of Computing*, 1993, 130 - 136.

[C] Z. Chen, "Learning unions of two rectangles in the plane with equivalence queries", *Proc. of the 6th Annual ACM Conference on Computational Learning Theory*, 1993, 243 - 252.

[CF] P. R. Cohen, E. A. Feigenbaum, "The Handbook of Artificial Intelligence" vol. 3, William Kaufmann (Los Altos, 1982).

[DGW] A. Dhagat, P. Gacs, P. Winkler, "On playing "Twenty Questions" with a liar", *Proc. of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, 1992, 16 - 22.

[H] D. Haussler, *personal communication*, 1989.

[L] N. Littlestone, "Learning quickly when irrelevant attributes abound: a new linear threshold algorithm", *Machine Learning*, 2, 1987, 285 - 318.

[MTa] W. Maass, G. Turán, "On the complexity of learning from counterexamples (extended abstract)", *Proc of the 30th Annual I.E.E.E. Symposium on Foundations of Computer Science*, 1989, 262 - 267.

[MTb] W. Maass, G. Turán, "Algorithms and lower bounds for on-line learning of geometrical concepts", Report 316 (Oct. 1991), *IIG-Report Series, Technische Universitaet Graz*; to appear in *Machine Learning.*

[MTc] W. Maass, G. Turán, "Lower bound methods and separation results for on-line learning models", *Machine Learning*, 9, 1992, 107 - 145.

[PV] L. Pitt, L. G. Valiant, "Computational limitations on learning from examples", *J. of the ACM,* 35, 1988, 965 - 984.

[S] R. I. Soare, "Recursively Enumerable Sets and Degrees", Springer (Berlin, 1987).

[V] L.G. Valiant, "A theory of the learnable", *Comm. of the ACM*, 27, 1984, 1134 - 1142.