Fast Identification of Geometric Objects with Membership Queries

WILLIAM J. BULTMAN

Department of Computer Science, University of Wisconsin Center Fox Valley, Menasha, Wisconsin 54952
E-mail: wbultman@uwcmail.uwc.edu

AND

WOLFGANG MAASS

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

We investigate the number of membership queries that are needed to identify polygons (i.e., intersections of halfplanes) over a two dimensional grid $\{0, ..., n-1\}^2$. We exhibit a learning algorithm that learns 100% correctly, while requiring no random examples and not more membership queries than previous algorithms needed in addition to their random examples for probably almost correctly learning (even for moderate values of ϵ, δ). Furthermore, the learning algorithm in this paper uses only grid points for its membership queries. This appears to be appropriate in situations where the probing device has only a limited resolution, and for applications to the set of pixels on a two-dimensional screen. The learning algorithm that is described in this paper overcomes a fundamental obstacle related to the zigzag borderline that is generated by a halfplane over a discrete grid. This obstacle had thwarted previous attempts to show that, in Angluin's model for on-line learning with equivalence and membership queries, one can learn in an efficient manner even those classes of geometric objects which cannot be learnt fast by equivalence queries alone (such as rectangles in general position). © 1995 Academic Press, Inc.

1. INTRODUCTION

We consider the complexity of learning with membership queries for three concept classes of geometric objects over the discrete 2-dimensional grid $\{0, ..., n-1\}^2$. A halfplane $\mathscr{H}_{\vec{\alpha}}(\vec{\alpha} \in \mathbb{Z}^3 - \{\mathbf{0}\})$ is the set of points above some line or the set of points below line, i.e.,

$$\mathscr{H}_{\vec{\alpha}} := \{\langle x, y\rangle \in \Re^2 \mid \alpha_1 x + \alpha_2 y \geqslant \alpha_3\}.$$

A rectangle $\mathscr{R}$ is an intersection of four halfplanes such that the four edges formed meet at right angles. Working in the grid, we consider the class of halfplanes (HALFSPACE$_n^2$), the class of polygons, or intersections of k many halfplanes (k-HALFPLANES$_n^2$), and its subclass of rectangles in general position (GP-BOX$_n^2 \subseteq$ 4-HALFSPACE$_n^2$), where

$$\text{HALFSPACE}_n^2 := \{\mathscr{H}_{\vec{\alpha}} \cap \{0, ..., n-1\}^2 \mid \vec{\alpha} \in \mathbb{Z}^3 - \{\mathbf{0}\}\},$$

$$k\text{-HALFSPACE}_n^2 := \{\mathbf{C}_1 \cap \cdots \cap \mathbf{C}_k \mid \mathbf{C}_1, ..., \mathbf{C}_k \in \text{HALFSPACE}_n^2\},$$

$$\text{GP-BOX}_n^2 := \{\mathscr{R} \cap \{0, ..., n-1\}^2 \mid \mathscr{R} \text{ is a rectangle}\}.$$

In this paper we investigate the complexity of learning (i.e., identifying) objects from these three classes in Angluin's model for on-line learning with equivalence and membership queries [Ang88]. A learning process in this model is viewed as a dialogue between the *learner* and the *environment*. The goal of the learner is to learn an unknown *target concept* $\mathbf{C}_T \in \mathscr{C}$ that has been fixed by the environment (one assumes that the concept class $\mathscr{C}$ and the considered domain X, with $\mathscr{C} \subseteq 2^X$, are known to both the learner and the environment). The learner may probe the environment with equivalence queries of the form "$\mathbf{H} = \mathbf{C}_T$?" for some $\mathbf{H} \in \mathscr{C}$. The environment responds to such a query with the reply "yes," or the reply "no" together with some *counterexample* $y \in (\mathbf{C}_T - \mathbf{H}) \cup (\mathbf{H} - \mathbf{C}_T)$. The learner may also ask membership queries "$x \in \mathbf{C}_T$?" for some $x \in X$, to which the environment responds with the reply "yes" or "no." The learning complexity LC-MEMB($\mathscr{A}$) of a learner (more precisely a learning algorithm) $\mathscr{A}$ is the maximal number of equivalence and membership queries that $\mathscr{A}$ needs in some learning process of this type before it can uniquely identify the target concept $\mathbf{C}_T$. The learning complexity LC-MEMB($\mathscr{C}$) of the concept class $\mathscr{C}$ is defined in this model by the minimum LC-MEMB($\mathscr{A}$) over all learning algorithms $\mathscr{A}$ which learn $\mathscr{C}$ using equivalence and membership queries. We write LC($\mathscr{C}$) (resp. MEMB($\mathscr{C}$)) for the corresponding learning complexities of $\mathscr{C}$ in the two more restricted models, where the learning algorithm $\mathscr{A}$ may only use equivalence queries (resp. only membership queries).

0890-5401/95 $6.00

It has been shown that $\mathrm{LC}(\textsc{Halfspace}_n^2) = \Theta(\log n)$ [MT89, MT91], whereas $\mathrm{LC}(2\text{-}\textsc{Halfspace}_n^2) = \Omega(n)$ and $\mathrm{LC}(\text{GP-Box}_n^2) = \Omega(n)$ [MT90, MT91b]. The latter results may be interpreted as saying that equivalence queries alone are insufficient for identifying intersection of halfplanes or rectangles in general position with a feasible number of queries.

Various other important concept classes for which equivalence queries alone are insufficient (such as DFAs [Ang87b, Ang89], k-term DNF [Ang87a, PV88], read-once formulas [AHK89], and conjunctions of Horn clauses [AFP90]) have been shown to be learnable in an efficient manner by a learning algorithm that employs both equivalence and membership queries. Furthermore, Baum [Bau90a, Bau90c] has recently shown that intersections of halfspaces become probably almost correctly learnable in an extended version of the PAC model where the learner may also ask membership queries, provided that the distribution and the target concept are chosen in "non-malicious manner." We will discuss the performance of Baum's algorithm for the learning problem considered here in Remark 4.4.

However, it has not been possible to show in Angluin's model for on-line learning that membership queries together with equivalence queries allow a faster identification of nontrivial geometric figures than equivalence queries alone. Apparently, this is due to a rather fundamental obstacle. The intersection of a non-axis-parallel geometric figure $\mathscr{B}$ in the Euclidean real plane $\mathfrak{R}^2$ with the underlying discrete grid $\{0, ..., n-1\}^2$ gives rise, in general, to a set $\mathscr{B} \cap \{0, ..., n-1\}^2$ of pixels whose boundary forms rather complicated *zigzag lines*. This effect arises even for very simple geometric objects such as halfplanes. No methods are available that would allow us to determine, with altogether $O(\log n)$ membership queries, on which side of the zigzag line every pixel lies (such a zigzag line has length $\Omega(n)$ in general).

There is one concept class $\mathscr{C}$ for which this on-line learning model has been able to determine quickly the exact zigzag borderline of arbitrary, not necessarily axis-parallel target objects from $\mathscr{C}$: the class $\textsc{Halfspace}_n^2$ [MT89, MT91a]. In this case, the employed reduction of the problem to the feasibility problem for convex sets in the dual space, in combination with powerful tools from combinatorial optimization, allow us to determine the exact zigzag borderline of an arbitrary halfplane with $O(\log n)$ equivalence queries. However, this method cannot be generalized to the learning of intersections of two or more halfspaces, since the latter problem has not been reduced to a tractable problem in the dual space. In fact, it has been shown that $\mathrm{LC}(2\text{-}\textsc{Halfspace}_n^2) = \Omega(n)$ [MT90, MT91b]. In the model LC-MEMB considered here, the learner may also use membership queries, but there has been no evidence so far that membership queries are of any additional use for determining a zigzag borderline quickly.

In this paper, we show that membership queries are in fact substantially more powerful than equivalence queries for on-line learning of nontrivial geometric figures. We solve the problem of designing fast learning algorithms in the LC-MEMB model for learning intersections of halfplanes in the following way. We first revisit the more fundamental problem of learning a single halfplane. In Section 2, we prove that $\mathrm{MEMB}(\textsc{Halfspace}_n^2) = \Theta(\log n)$. The algorithm that we introduce to prove the upper bound of this result is based on a completely different approach than the one used by [MT89] to show that $\mathrm{LC}(\textsc{Halfspace}_n^2) = O(\log n)$. We show that the zigzag line problem is essentially the only problem in learning the class $\textsc{Halfspace}_n^2$. We do this by reducing $\textsc{Halfspace}_n^2$ to the concept class $\textsc{Strip}_{X,Y}$, the embodiment of the zigzag line problem. We then present a recursive learning algorithm for $\textsc{Strip}_{X,Y}$. The new algorithm has the advantage that it can also be used as a subroutine for the learning of more complex geometric figures.

While zigzag lines are not the only problem for learning the classes $k\text{-}\textsc{Halfspace}_n^2$ and GP-Box_n^2, they are a major part of the problem. In Section 3, we develop methods for tracing along the edges of a polygon to approximately locate its vertices. Since the vertices of a polygon occur at real-valued points, such approximate location, to within a small convex region, is all we can hope for. Our design of the edge tracing procedure employs a method which we call *inverse binary search*. This search proceeds in up to $\log n$ phases. In each phase it either determines a new border pair twice as far along the traced edge $\mathscr{E}$ as the previous border pair, or finds evidence that the edge $\mathscr{E}$ ends shortly after this border pair. Inverse binary search can be carried out with $O(\log n)$ membership queries, since each of its phases requires only two membership queries. This is possible in spite of the exponentially increasing distance between the successively determined border pairs for $\mathscr{E}$, since the remaining uncertainty about the slope of $\mathscr{E}$ decreases exponentially fast from phase to phase.

Once two adjacent vertices are approximately located by this tracing procedure, we can run the learning algorithm for $\textsc{Strip}_{X,Y}$ to determine the points on the zigzag border line of the edge between them. In this way, we develop a fast learning algorithm for $k\text{-}\textsc{Halfspace}_n^2$ (under certain restrictions). In Section 4, we consider the "natural" subclasses of $k\text{-}\textsc{Halfspace}_n^2$ whose angles are bounded away from 0 and π by some fixed constant ρ. The class GP-Box_n^2 is one such class.

2. LEARNING OF A HALFPLANE WITH MEMBERSHIP QUERIES

In this section we show that for any given halfplane $\mathscr{H}_\mathrm{T} \subseteq \mathfrak{R}^2$ one can determine the precise set $\mathscr{H}_\mathrm{T} \cap \{0, ..., n-1\}^2$ (in particular its zigzag borderline) with $O(\log n)$ queries "$\mathbf{p} \in \mathscr{H}_\mathrm{T}$?" for $\mathbf{p} \in \{0, ..., n-1\}^2$. The learning algorithm that we design provides an essential subroutine for the on-line learning algorithms for more complex geometrical figures in Section 3.

THEOREM 2.1.

$$\text{MEMB}(\text{HALFSPACE}_n^2) = \Theta(\log n).$$

Furthermore, there is a learning algorithm for HALFSPACE$_n^2$ *that uses at most* $O(\log n)$ *membership queries and, after polynomially in* $\log n$ *many computation steps, outputs the coefficients* $\vec{\alpha} \in \mathbb{Z}^3 - \{\mathbf{0}\}$ *of some separating halfplane* $\mathcal{H}_{\vec{\alpha}}$ *(i.e.,* $\mathcal{H}_{\vec{\alpha}} \cap \{0, ..., n-1\}^2 = \mathbf{C}_\mathrm{T}$*).*

Proof. The lower bound follows from the general fact that $\text{MEMB}(\mathscr{C}) \geq \log |\mathscr{C}|$ for any concept class $\mathscr{C}$.

For the proof of the upper bound we assume that the environment has fixed some target concept $\mathbf{C}_\mathrm{T} \in$ HALFSPACE$_n^2$. When necessary, a point will be represented by its x- and y-components $\mathbf{p} = \langle p_x, p_y \rangle$. Arithmetic operations and relations on points are always spread componentwise; i.e., for $\mathbf{p} = \langle p_x, p_y \rangle$, $\mathbf{q} = \langle q_x, q_y \rangle$ and $r \in \mathfrak{R}$,

$$\begin{aligned} \mathbf{p} \pm \mathbf{q} &:= \langle p_x \pm q_x, p_y \pm q_y \rangle, \\ r\mathbf{p} &:= \langle rp_x, rp_y \rangle, \\ |\mathbf{p}| &:= \langle |p_x|, |p_y| \rangle, \\ \mathbf{p} \leq \mathbf{q} &\Leftrightarrow p_x \leq q_x \quad \text{and} \quad p_y \leq q_y. \end{aligned}$$

Grid points $\mathbf{p}_1, ..., \mathbf{p}_m$ are *rowwise adjacent* if $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ lie in the same column and adjacent rows for $i = 1, ..., m-1$. In this case we call the point set $\{\mathbf{p}_1, ..., \mathbf{p}_m\}$ rowwise adjacent. Two grid points $\mathbf{p}$, $\mathbf{q}$ are called a rowwise *border pair* $[\mathbf{p}, \mathbf{q}]$ if $\mathbf{p} \in \mathbf{C}_\mathrm{T}$, $\mathbf{q} \notin \mathbf{C}_\mathrm{T}$, and they are rowwise adjacent. Similar definitions hold for columnwise border pair. The convex hull of the points $\mathbf{p}_1, ..., \mathbf{p}_m$ is defined as

$$\text{conv}\{\mathbf{p}_1, ..., \mathbf{p}_m\} := \left\{ \sum_{i=1}^{m} \alpha_i \mathbf{p}_i \mid \alpha_1, ..., \alpha_m \geq 0, \sum_{i=1}^{m} \alpha_i = 1 \right\}.$$

It is obvious that, unless $\mathbf{C}_\mathrm{T} = \{0, ..., n-1\}^2$ or $\mathbf{C}_\mathrm{T} = \phi$, at least one of the corner points of the grid belongs to $\mathbf{C}_\mathrm{T}$ and one does not. Hence, one can determine by binary search, with $O(\log n)$ membership queries, the two border pairs that lie on the perimeter of the grid $\{0, ..., n-1\}^2$. One might believe that $\mathbf{C}_\mathrm{T}$ is uniquely determined by these border pairs (this error led to the premature statement of the result of this theorem in [MT89]). However, it is obvious that the determination of all border pairs on the perimeter leaves a set $\mathbf{U}$ of up to $n-2$ grid points inside the grid unclassified (see Fig. 1).

The determination of $\mathbf{U} \cap \mathbf{C}_\mathrm{T}$ is a special case of a general problem that arises whenever one wants to identify non-axis-parallel geometric figures over a fixed grid. These

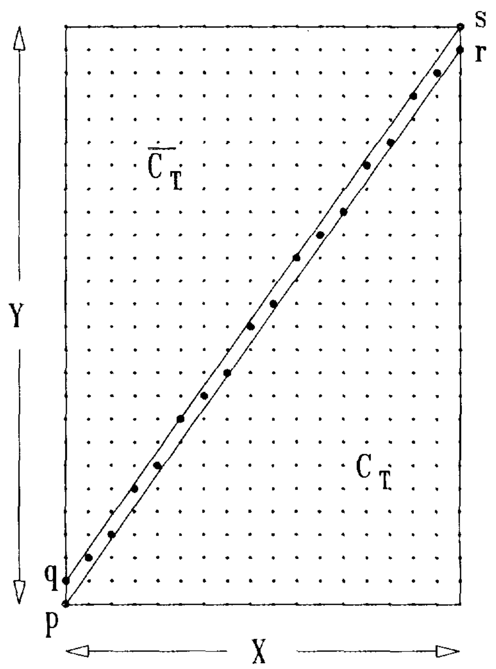


FIG. 1. The set of unclassified grid points in STRIP$_{X,Y}$.

appears to be no easy solution to this problem, since the set $\mathbf{U}$ may have a rather complicated geometric structure (see Fig. 1).

We solve the problem of determining $\mathbf{U} \cap \mathbf{C}_\mathrm{T}$ by showing that, with the help of results from the theory of rational approximation, one can construct an affine transformation $\mathscr{A}$ that maps the points in $\mathbf{U}$ into a set $\mathscr{A}[\mathbf{U}]$ of grid points in a substantially smaller new grid. The new grid is the skew grid which is indicated in Fig. 3. Our learning algorithm will proceed in a recursive fashion, by determining with $O(\log n)$ membership queries the border pairs for $\mathscr{A}[\mathbf{C}_\mathrm{T}]$ on the perimeter of $\mathscr{A}[\mathbf{U}]$, thereby reducing the problem of determining $\mathbf{U} \cap \mathbf{C}_\mathrm{T}$ to a similar problem over a smaller grid. Some technical complications arise, since many grid points in the new grid do not correspond to grid points in the original grid. Obviously, such points may not be used for membership queries in our algorithm.

We define a class that contains all the difficulty of the zigzag line problem for HALFSPACE$_n^2$ i.e., the determination of $\mathbf{U} \cap \mathbf{C}_\mathrm{T}$. This class is defined on the grid $\{0, ..., X\} \times \{0, ..., Y\}$. The class STRIP$_{X,Y}$ contains all halfplanes that run from the lower-left corner to the upper right corner of the grid, i.e.,

$$\begin{aligned} \text{STRIP}_{X,Y} := \{ \mathcal{H}_{\vec{\alpha}} \cap (\{0, ..., X\} \times \{0, ..., Y\}) \mid \vec{\alpha} \\ \in \mathbb{Z}^3 - \{\mathbf{0}\}, \langle 0, 0 \rangle \in \mathcal{H}_{\vec{\alpha}}, \\ \langle 0, 1 \rangle \notin \mathcal{H}_{\vec{\alpha}}, \\ \langle X, Y-1 \rangle \in \mathcal{H}_{\vec{\alpha}}, \\ \text{and } \langle X, Y \rangle \notin \mathcal{H}_{\vec{\alpha}} \}. \end{aligned}$$

The main statement of Theorem 2.1 follows from the following two lemmas. We will discuss the computation time of the algorithm at the end of Section 2.

LEMMA 2.2.

$$\mathrm{MEMB}(\textsc{Halfspace}_n^2) \leqslant 2\lceil \log n \rceil + 4 + \max_{\substack{X \leqslant n-1 \\ Y \leqslant n-1}} \{\mathrm{MEMB}(\textsc{Strip}_{X,Y})\}.$$

Proof. We begin by querying each of the four corners of the grid: $\langle 0, 0\rangle$, $\langle 0, n-1\rangle$, $\langle n-1, 0\rangle$, and $\langle n-1, n-1\rangle$. If all four corner points are in $\mathbf{C}_\mathrm{T}$, then $\mathbf{C}_\mathrm{T} = \{0, ..., n-1\}^2$, and we are done. Similarly, if all four corner points are out of $\mathbf{C}_\mathrm{T}$, then $\mathbf{C}_\mathrm{T} = \phi$, and we are done. Thus, we may assume that at least one corner point is in $\mathbf{C}_\mathrm{T}$ and one corner point is out of $\mathbf{C}_\mathrm{T}$. From this it follows that we know exactly two sides of the grid whose endpoints (corner points) differ as to membership in $\mathbf{C}_\mathrm{T}$. We wish to find a border pair on both of those sides.

Given a side of the grid with one end point $\mathbf{p} \in \mathbf{C}_\mathrm{T}$ and the other end point $\mathbf{q} \notin \mathbf{C}_\mathrm{T}$, we can find a border pair on that side with at most $\lceil \log_2(n-1) \rceil$ queries, using binary search. We begin by probing the point $\mathbf{t} = \lceil \frac{1}{2}(\mathbf{p} + \mathbf{q}) \rceil$. If $\mathbf{t} \in \mathbf{C}_\mathrm{T}$, we set $\mathbf{p} \leftarrow \mathbf{t}$ and repeat. If $\mathbf{t} \notin \mathbf{C}_\mathrm{T}$, we set $\mathbf{q} \leftarrow \mathbf{t}$ and repeat. When $\mathbf{p}$ and $\mathbf{q}$ are adjacent, we have found a border pair. Similarly, we can find a border pair $[\mathbf{r}, \mathbf{s}]$ on the other grid side.

The only unclassified points in the grid lie between the border pairs $[\mathbf{p}, \mathbf{q}]$ and $[\mathbf{r}, \mathbf{s}]$ (see Fig. 2). All that remains is to transform the grid to "look like" an instance of $\textsc{Strip}_{X,Y}$ (see Fig. 1). That is, we want both border pairs columnwise in the lower-left and upper-right corners of the grid.

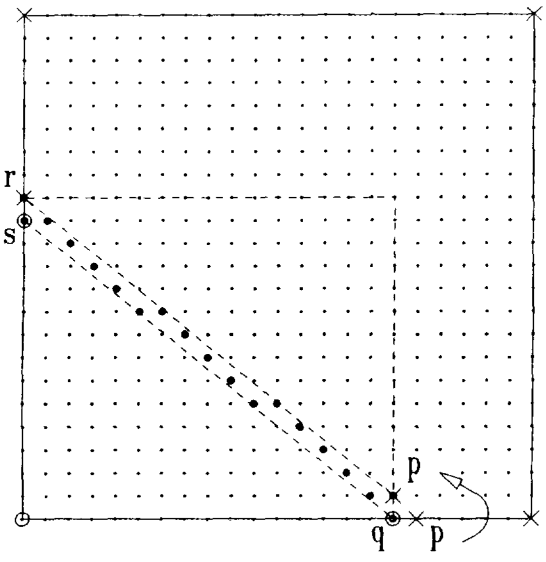


FIG. 2. An instance of $\textsc{Strip}_{X,Y}$ in the input grid of $\textsc{Halfspace}_n^2$.

We begin by making the two border pairs $[\mathbf{p}, \mathbf{q}]$ and $[\mathbf{r}, \mathbf{s}]$ either both rowwise adjacent or both columnwise adjacent, i.e., making $\mathbf{p} - \mathbf{q} = \mathbf{r} - \mathbf{s}$. This is already true if the border pairs are on opposite sides of the grid, but requires another step if they are no adjacent sides. We must replace the point among $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$, and $\mathbf{s}$ that is furthest from the common endpoint (corner) of the adjacent sides. For example, if this point is $\mathbf{p}$, then we replace $\mathbf{p}$ with the unique neighbor of $\mathbf{q}$ that is in the interior of the grid (see Fig. 2). In general, we replace one point of a border pair with the unique interior neighbor of the other point of the border pair (the special case when there is no such interior neighbor results only from a trivial concept that contains or excludes exactly one grid point). The replacement point is guaranteed to have the same membership property as the point that it replaces.

At this point, we have almost completed the reduction. We wish to concentrate on the minimal rectangular subgrid $\tilde{\mathbf{G}}$ containing $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$, and $\mathbf{s}$. By convexity of halfplanes, the membership in $\mathbf{C}_\mathrm{T}$ for all grid points outside $\tilde{\mathbf{G}}$ is known. It is easy to see that $\tilde{\mathbf{G}}$ is isomorphic to an instance of $\textsc{Strip}_{X,Y}$ for some $X, Y \leqslant n-1$, by flipping and/or rotating the grid. More precisely, there is some affine transformation $\mathscr{A}$ of $\mathfrak{R}^2$ such that $\mathscr{A}[\tilde{\mathbf{G}}] \in \textsc{Strip}_{X,Y}$. Running a learning algorithm for $\textsc{Strip}_{X,Y}$ on the $X \times Y$ subgrid $\mathscr{A}[\tilde{\mathbf{G}}]$ will determine the membership in $\mathbf{C}_\mathrm{T}$ for all points in the original grid $\{0, ..., n-1\}^2$.

As a practical matter of implementation, we do not actually transform the grid. Rather, we transform the queries made on the grid. When a membership query "$\mathbf{p} \in \mathscr{A}[\mathbf{C}_\mathrm{T}]$?" would be made on a grid to which a transformation $\mathscr{A}$ has been applied, the algorithm actually makes the query "$\mathscr{A}^{-1}(\mathbf{p}) \in \mathbf{C}_\mathrm{T}$?" on the original grid. This is necessary, as the oracle for membership queries can only answer queries to the original grid.

We will be using affine transformations of the grid extensively in the following proofs. Since affine transformations preserve linearity, we will always be attempting to learn some (transformed) halfplane in some (transformed) grid. Each time a new transformation is applied to the grid, we compose it with those transformations applied so far. This allows us to talk rather loosely about making a membership query in any transformed grid, letting some overseeing watchdog always apply the inverse of the composed transformations to each point before actually querying.

The only membership queries required by this reduction are of the four corner points, and those for two binary searches, a total of at most $2\lceil \log n \rceil + 4$. This completes the proof of Lemma 2.2. ∎

LEMMA 2.3.

$$\mathrm{MEMB}(\textsc{Strip}_{X,Y}) \leqslant 4 \log X + 33\lceil \log \log X \rceil - 30.$$

It immediately follows from Lemmas 2.2 and 2.3 that

$$\mathrm{MEMB}(\textsc{Halfspace}_n^2) \leqslant 6\lceil \log n \rceil + 33\lceil \log\log n \rceil - 26.$$

The small constants make this a practical algorithm for relatively small values of n, when the cost of a query is a limiting factor of an algorithm. For example, learning an arbitrary halfplane on a megapixel (1024×1024) display requires querying in the worst case 113 pixels, or about 1 in every 9000 pixels.

Proof. (of Lemma 2.3). We reduce $\textsc{Strip}_{X,Y}$ to $\textsc{Strip}_{X',Y'}$ on a smaller grid. The following lemma gives the result of that reduction. The result of Lemma 2.3 follows from the solution to the recurrence implicit in the following lemma, and a trivial basis case.

LEMMA 2.4.

$$\mathrm{MEMB}(\textsc{Strip}_{X,Y}) \leqslant \lceil \log X \rceil + 31 + \max_{\substack{X' \leqslant 2\sqrt{X} \\ Y' \leqslant X + 2\sqrt{X}}} \{\mathrm{MEMB}(\textsc{Strip}_{X',Y'})\}.$$

The proof of Lemma 2.4 will follow. We continue with the proof of Lemma 2.3.

By probing all of the at most $X-1$ points in the set of unclassified points $\mathbf{U} = \mathrm{conv}\{\mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s}\} \cap (\{0, ..., X\} \times \{0, ..., Y\})$ of $\textsc{Strip}_{X,Y}$, we have the basis case

$$\mathrm{MEMB}(\textsc{Strip}_{X,Y}) \leqslant X - 1.$$

From this, and the result of Lemma 2.4, it is clear that $\mathrm{MEMB}(\textsc{Strip}_{X,Y})$ can be bounded by a function dependent only on X, not Y. Let us define a recurrence variable

$$S(m) := \max_{0 \leqslant X \leqslant 2^{2^m}+2} \{\mathrm{MEMB}(\textsc{Strip}_{X,Y})\}.$$

Then, by Lemma 2.4, we have

$$S(m) \leqslant S(m-1) + \log(2^{2^m+2}) + 31$$
$$= S(m-1) + 2^m + 33.$$

From the basis case we have

$$S(1) = \max_{0 \leqslant X \leqslant 16} \{\mathrm{MEMB}(\textsc{Strip}_{X,Y})\}$$
$$\leqslant 15.$$

A bound on this recurrence is

$$S(m) \leqslant 2^{m+1} + 33m - 22,$$

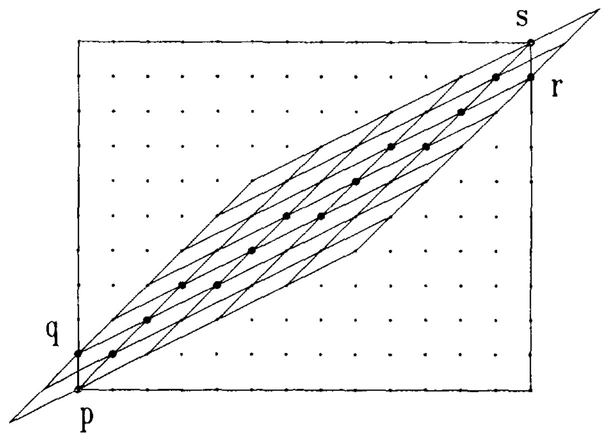


FIG. 3. A skew grid within the original grid.

from which it follows that

$$\mathrm{MEMB}(\textsc{Strip}_{X,Y}) \leqslant S(\lceil \log\log \tfrac{1}{4} X \rceil)$$
$$\leqslant 4 \log \tfrac{1}{4} X +$$
$$+ 33\lceil \log\log \tfrac{1}{4} X \rceil - 22$$
$$\leqslant 4 \log X +$$
$$+ 33\lceil \log\log X \rceil - 30.$$

This completes the proof of Lemma 2.3. ∎

Proof. (of Lemma 2.4). The concepts we now consider from $\textsc{Strip}_{X,Y}$. Thus, the halfplane $\mathscr{H}_{\mathrm{T}}$ and target concept $\mathbf{C}_{\mathrm{T}} = \mathscr{H}_{\mathrm{T}} \cap (\{0, ..., X\} \times \{0, ..., Y\})$ which we consider for the remainder of this proof will in general be different from the original halfplane and target concept. If $X = 0$ or $Y = 0$ the problem is trivial, so we may assume that $X, Y > 0$. We reduce our current problem to a problem on a skew grid, embedded in the current grid and containing the strip of unclassified points $\mathbf{U}$ (see Fig. 3[1]). The two sets of parallel lines defining the skew grid are chosen by their slopes a/b and c/d ($a, b, c, d \in \mathbb{Z}$; $a, c \geqslant 0$; $b, d > 0$) respectively.

In order for the reduction to go through, the new grid must satisfy two properties. First, the size of the new grid must decrease significantly in the x-dimension. The first set of lines of slope a/b will be chosen to satisfy this property. Second, the grid points (intersections of grid lines) of the new grid must correspond exactly to the grid points of the original grid. The second set of lines of slope c/d will be chosen to satisfy this property.

Let $\mathbf{p} := \langle 0, 0 \rangle$, $\mathbf{q} := \langle 0, 1 \rangle$, $\mathbf{r} := \langle X, Y-1 \rangle$ and $\mathbf{s} := \langle X, Y \rangle$ as in Fig. 3. If a and b are relatively prime, the lines

[1] The skew grid in Fig. 3 is *not* the one constructed by the method in the text. The skew grids actually constructed tend to be so long and thin as to be nearly unrecognizable in a diagram.

of slope a/b that intersect integer valued points are the lines

$$y = \frac{a}{b}x + \gamma\frac{1}{b} \quad (\gamma \in \mathbb{Z}).$$

The lines that intersect conv$\{\mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s}\}$ are exactly those that intersect the line segment $\overline{\mathbf{pr}}$ plus those that intersect one of the border pair segments ($\overline{\mathbf{pq}}$ if $a/b > (Y-1)/X$: $\overline{\mathbf{rs}}$ otherwise). The number of such lines is $\tilde{X} + 1$, where

$$\begin{aligned}\tilde{X} &= |b(Y-1) - aX| + |b| \\ &= bX\left|\frac{a}{b} - \frac{Y-1}{X}\right| + b.\end{aligned}$$

To minimizing this, we must simultaneously minimize both b and $|a/b - (Y-1)/X|$. That is, we must closely approximate $(Y-1)/X$ by a fraction with a small denominator. We use the following lemma from the theory of rational approximation to accomplish this (see Lov86, Hua82).

Lemma 2.5. *For all $\alpha \in \Re$ and $\beta \in \mathbb{N}^+$ there exist $a, b \in \mathbb{Z}$ such that $1 \leqslant b \leqslant \beta$ and*

$$\left|\frac{a}{b} - \alpha\right| \leqslant \frac{1}{b(\beta+1)}.$$

Clearly we may also choose a and b so that $\gcd(a, b) = 1$. We will assume that $a/b \geqslant (Y-1)/X$; the other case is handled analogously. Applying Lemma 2.5 with $\alpha = (Y-1)/X$ and $\beta = \lfloor\sqrt{X}\rfloor$ we have

$$1 \leqslant b \leqslant \sqrt{X},$$

$$\left|\frac{a}{b} - \frac{Y-1}{X}\right| < \frac{1}{b\sqrt{X}},$$

from which it follows that

$$\begin{aligned}\tilde{X} &= bX\left|\frac{a}{b} - \frac{Y-1}{X}\right| + b \\ &< bX\frac{1}{b\sqrt{X}} + \sqrt{X} \\ &= 2\sqrt{X}.\end{aligned}$$

Now we choose set the second set of lines of slope c/d. In order that the new grip points correspond exactly to the original grid points, we must choose c/d to be the slope of a line between two points on adjacent lines in the first set. A good choice is to choose two "neighbor" points, that is, a point on some line $y = (a/b)x + \gamma(1/b)$ and some point on one of the two adjacent lines $y = (a/b)x + (\gamma + \delta)(1/b)$ where ($\delta = \pm 1$). We choose which adjacent line based on the sign of our error in approximating $(Y-1)/X$ by a/b. That is, define

$$\delta = \begin{cases} +1, & \text{if } \dfrac{Y-1}{X} - \dfrac{a}{b} \leqslant 0; \\ -1, & \text{otherwise.}\end{cases}$$

From the constraints above, we determine that we would choose d to be the unique integer such that

$$ad + \delta \equiv 0 \pmod{b} \quad (1 \leqslant d \leqslant b)$$

(if $b = 1$ let $d = 1$). Such a number d exists since $\gcd(a, b) = 1$. We then set

$$c := \frac{ad + \delta}{b} \in \mathbb{N}.$$

Now, since

$$\begin{aligned}\gcd(bc, d) &= \gcd(ad + \delta, d) \\ &= \gcd(\delta, d) \\ &= 1,\end{aligned}$$

it follows that $\gcd(c, d) = 1$. Thus, the lines of the second set are all lines

$$y = \frac{c}{d}x + \gamma\frac{1}{d} \quad (\gamma \in \mathbb{Z})$$

that intersect conv$\{\mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s}\}$. The number of such lines is $\tilde{Y} + 1$, where

$$\begin{aligned}\tilde{Y} &= |d(Y-1) - cX| + |d| \\ &= \left|d(Y-1) - \frac{ad+\delta}{b}X\right| + |d| \\ &= dX\left|\frac{Y-1}{X} - \frac{a}{b} - \frac{\delta}{bd}\right| + |d| \\ &= dX\left|\frac{Y-1}{X} - \frac{a}{b}\right| + dX\left|\frac{\delta}{bd}\right| + |d| \\ &= \frac{d}{b}\left(bX\left|\frac{Y-1}{X} - \frac{a}{b}\right| + |b|\right) + dX\frac{1}{bd} \\ &= \frac{d}{b}\tilde{X} + \frac{1}{b}X \\ &\leqslant X + \tilde{X} \\ &< X + 2\sqrt{X}.\end{aligned}$$

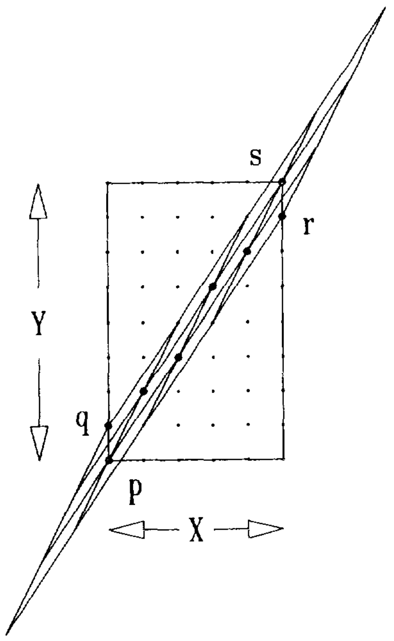


FIG. 4. The skew grid within the original grid for $\text{STRIP}_{5,8}$.

We then apply the affine transformation

$$\mathscr{A}\colon \langle u, v\rangle \mapsto \langle au + b(1-v), cu + d(1-v)\rangle$$

to the skew grid to make it orthogonal, with its lower-left corner $\mathscr{A}(\mathbf{q})$ at the origin. Applying this construction to the grid in Fig. 4 results in the new grid in Fig. 5. By the previous computation, the new grid is $\{0, ..., \tilde{X}\} \times \{0, ..., \tilde{Y}\}$ where $\tilde{X} < 2\sqrt{X}$ and $\tilde{Y} < X + 2\sqrt{X}$. What remains is to learn how the halfplane $\mathscr{A}[\mathscr{H}_T]$ intersects this new grid.

Unfortunately, we cannot proceed as in Lemma 2.2 by searching along the sides of the new grid. The points along the sides of the new grid do not, in general, correspond to points within the original grid (see Figs. 4 and 5). Fortunately, as we have seen before, the only unclassified grid points are those in $\mathscr{A}[\mathbf{U}]$, whch are contained in $\mathscr{A}[\text{conv}\{\mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s}\}] = \text{conv}\{\mathscr{A}(\mathbf{p}), \mathscr{A}(\mathbf{q}), \mathscr{A}(\mathbf{r}), \mathscr{A}(\mathbf{s})\}$. We thus search for two border pairs along the "boundaries" of $\mathscr{A}[\mathbf{U}]$. Since $\mathscr{A}(\mathbf{p}), \mathscr{A}(\mathbf{r}) \in \mathscr{A}[\mathbf{C}_T]$ and $\mathscr{A}(\mathbf{q}), \mathscr{A}(\mathbf{s}) \notin \mathscr{A}[\mathbf{C}_T]$, it is enough to search the boundary of $\mathscr{A}[\mathbf{U}]$ between $\mathscr{A}(\mathbf{p})$ and $\mathscr{A}(\mathbf{q})$, and the boundary of $\mathscr{A}[\mathbf{U}]$ between $\mathscr{A}(\mathbf{r})$ and $\mathscr{A}(\mathbf{s})$.

These "boundaries" are the zigzag lines $\mathscr{Z}^+_{p,q}$ (resp. $\mathscr{Z}^-_{r,s}$) that run parallel to and above (resp. below) the line segment $\overline{\mathscr{A}(\mathbf{p})\,\mathscr{A}(\mathbf{q})}$ (resp. $\overline{\mathscr{A}(\mathbf{r})\,\mathscr{A}(\mathbf{s})}$). Since $\mathscr{A}(\mathbf{q}) = \langle 0, 0\rangle$ and $\mathscr{A}(\mathbf{p}) = \langle b, d\rangle$, the discrete grid points along these zigzag lines are, respectively,

$$\mathbf{Z}^+_{p,q} := \{\langle u, v\rangle \in \{0, ..., b\} \times \{0, ..., d\} \mid du \leqslant bv < du + (b+d)\}$$

$$\mathbf{Z}^-_{r,s} := \{\langle \tilde{X} - u, \tilde{Y} - v\rangle \in \{0, ..., b\} \times \{0, ..., d\} \mid du \leqslant bv < du + (b+d)\}.$$

The continuous zigzag lines $\mathscr{Z}^+_{p,q}$ and $\mathscr{Z}^-_{r,s}$ are formed by joining nearest neighbors of the corresponding discrete sets $\mathbf{Z}^+_{p,q}$ and $\mathbf{Z}^-_{r,s}$.

We want to determine $\mathscr{A}[\mathscr{H}_T] \cap \mathbf{Z}^+_{p,p}$. The problem is that the slope of the border line of $\mathscr{A}[\mathscr{H}_T]$ may be close to the slope of $\overline{\mathscr{A}(\mathbf{p})\,\mathscr{A}(\mathbf{q})}$ (see Fig. 6). Hence, this border line may cut the zigzag line $\mathscr{Z}^+_{p,q}$ in a rather complicated fashion, giving rise to many border pairs in $\mathbf{Z}^+_{p,q}$. The following lemma will imply that the intersection of the border line of $\mathscr{A}[\mathscr{H}_T]$ with $\mathscr{Z}^+_{p,q}$ cannot be very complicated, although it may give rise to several border pairs in $\mathbf{Z}^+_{p,q}$ (as indicated in Fig. 6).

LEMMA 2.6. *All intersections of the border line of $\mathscr{A}[\mathscr{H}_T]$ with $\mathscr{Z}^+_{p,q}$ (resp. $\mathscr{Z}^-_{r,s}$) occur within an x-distance of less than 4 units and a y-distance of less than 6 units from each other.*

Proof. We prove the lemma for $\mathscr{Z}^+_{p,q}$. The case of $\mathscr{Z}^-_{r,s}$ is analogous. Note that if the slope of the border line of $\mathscr{A}[\mathscr{H}_T]$ is negative, then there is exactly one intersection, and we are done. Thus, we assume that the slope of the border line is positive.

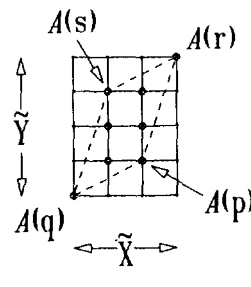


FIG. 5. The new orthogonal grid resulting from $\text{STRIP}_{5,8}$.

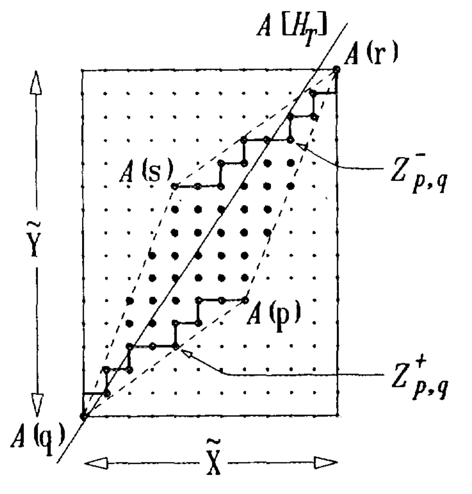


FIG. 6. The new orthogonal grid, with zigzag lines.

The idea of this proof is as follows. The slope of the border line of $\mathscr{A}[\mathscr{H}_T]$ is bounded below by $\tilde{Y}/\tilde{X}$, since it passes beween $\mathscr{A}(\mathbf{q}) = \langle 0, \rangle$ and $\mathscr{A}(\mathbf{r}) = \langle \tilde{X}, \tilde{Y}\rangle$. By definition, $\mathbf{Z}^+_{p,q}$ and therefore $\mathscr{Z}^+_{p,q}$ are sandwiched between the lines $y = (d/b)\,x$ and $y = (d/b)\,x + (1 + d/b)$, separated by a vertical distance of $(1 + d/b) \leqslant 2$. Since the slope of the border line of $\mathscr{A}[\mathscr{H}_T]$, $\tilde{Y}/\tilde{X}$, is significantly larger than the slope of the lines sandwiching $\mathscr{Z}^+_{p,q}$, d/b, the border line cannot remain between these lines for long.

Let us begin by showing that $\tilde{Y}/\tilde{X}$ is significantly larger than d/b. In the computation above, we saw that

$$\begin{aligned}\frac{\tilde{Y}}{\tilde{X}} &= \frac{(d/b)\,\tilde{X} + (1/b)\,X}{\tilde{X}}\\ &= \frac{d}{b} + \frac{X}{b\tilde{X}}\\ &> \frac{d}{b} + \frac{X}{b \cdot 2\sqrt{X}}\\ &= \frac{d}{b} + \frac{\sqrt{X}}{2b}.\end{aligned}$$

Note that, since $d \leqslant \sqrt{X}$, this shows that $\tilde{Y}/\tilde{X} > (3/2\, d/b)$, but we will need the stronger result as stated above.

Now let us consider how long the border line of $\mathscr{A}[\mathscr{H}_T]$ can remain between the lines sandwiching $\mathscr{Z}^+_{p,q}$. Let $\mathbf{t}_1$ (resp. $\mathbf{t}_2$) be the point of intersection of the borderline with $y = (d/b)\,x$ (resp. $y = (d/b)\,x + (1 + (d/b))$). Let $\langle l, h\rangle := \mathbf{t}_2 - \mathbf{t}_1$. To prove Lemma 2.6, we need to show that $l < 4$ and $h < 6$. It is easy to derive the following inequalities from the above discussion (see Fig. 7):

$$\frac{h}{l} > \frac{\tilde{Y}}{\tilde{X}},$$

$$\frac{h - (1 + d/b)}{l} = \frac{d}{b}.$$

Rewriting the above equation, we have

$$\begin{aligned} l &= \frac{1 + d/b}{h/l - d/b}\\ &< \frac{1 + d/b}{\tilde{Y}/\tilde{X} - d/b}\\ &< \frac{1 + d/b}{(d/b + \sqrt{X}/2b) - d/b}\\ &= \frac{2b + 2d}{\sqrt{X}}\\ &\leqslant \frac{2\sqrt{X} + 2\sqrt{X}}{\sqrt{X}}\\ &= 4.\end{aligned}$$

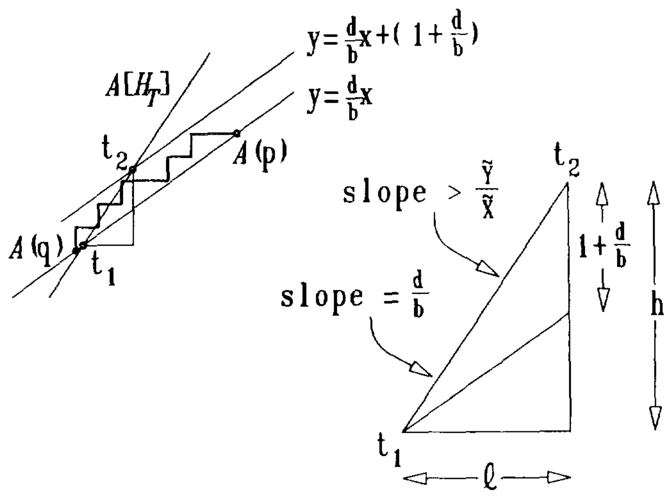


FIG. 7. The intersection between a zigzag line and the border line of a halfplane.

Rewriting the equation again, we have

$$\begin{aligned} h &= \frac{d}{b}\, l + \left(1 + \frac{d}{b}\right)\\ &\leqslant l + (1 + 1)\\ &< 4 + (1 + 1)\\ &= 6.\end{aligned}$$

This completes the proof of Lemma 2.6. ▌

We order the points in $\mathbf{Z}^+_{p,q}$ according to their order of occurrence along $\mathscr{Z}^+_{p,q}$. Since $\mathscr{A}(\mathbf{p}) \in \mathscr{A}[\mathbf{C}_T]$ and $\mathscr{A}(\mathbf{q}) \notin \mathscr{A}[\mathbf{C}_T]$ we can do a binary search along $\mathscr{Z}^+_{p,q}$ to find some border pair in $\mathbf{Z}^+_{p,q}$. By Lemma 2.6, all other border pairs occur within x-distance 4 and y-distance 6 from this discovered border pair. It is not difficult to see that there are at most $2 \cdot 2 \cdot 4 - 2 = 14$ grid points (two per column to the left and right of the border pair found) in $\mathbf{Z}^+_{p,q}$ that are left unclassified. We query these 14 unclassified points.

Let $[\bar{\mathbf{p}}, \bar{\mathbf{q}}]$ be the innermost (i.e., closest to $\mathscr{A}(\mathbf{p})$) border pair in $\mathbf{Z}^+_{p,q}$, and $[\bar{\mathbf{r}}, \bar{\mathbf{s}}]$ be the innermost (i.e., closest to $\mathscr{A}(\mathbf{s})$) border pair in $\mathbf{Z}^-_{r,s}$, found in a similar manner. We are now in a situation similar to that in Lemma 2.2. The only unclassified points in the grid lie between the border pairs $[\bar{\mathbf{p}}, \bar{\mathbf{q}}]$ and $[\bar{\mathbf{r}}, \bar{\mathbf{s}}]$. All that remains is to transform the grid to "look like" an instance of $\textsc{Strip}_{X', Y'}$. First, as in Lemma 2.2, we replace one of $\bar{\mathbf{p}}$, $\bar{\mathbf{q}}$, $\bar{\mathbf{r}}$ or $\bar{\mathbf{s}}$ so that both border pairs are rowwise or both are columnwise. Second, we apply another affine transformation (if needed) to rotate and/or flip the grid into the correct orientation.

This completes the reduction of $\textsc{Strip}_{X,Y}$ to $\textsc{Strip}_{X',Y'}$, where $X' \leqslant \tilde{X} \leqslant 2\sqrt{X}$ and $Y' \leqslant \tilde{Y} \leqslant X + 2\sqrt{X}$. The only membership queries used to do this were those for the 2 binary searches and the 14 remaining unclassified points on the zigzag line, a total of

$$\begin{aligned}2(\lceil \log_2(b+d)\rceil + 14) &\leqslant 2(\lceil \log 2\sqrt{X}\rceil + 14)\\ &= 2\lceil \tfrac{1}{2}\log X\rceil + 30\\ &\leqslant \lceil \log X\rceil + 31.\end{aligned}$$

We have reduced the detemination of $\mathbf{C}_\mathrm{T}$ to the determination of $\mathscr{A}[\mathbf{C}_\mathrm{T}]$. This completes the proof of Lemma 2.4. ∎

All that remains of the proof of Theorem 2.1 is to discuss the number of *computation* steps required by the above algorithm. All computations performed are on integers that are bounded by a polynomial in n, so we may use any standard arithmetic operation as an atomic operation. There are five areas where it is not trivial to see that the computation time is polylogarithmic in n: performing binary search; performing an affine transformation; choosing a/b; choosing c/d; and computing $\vec{\alpha} \in \mathbb{Z}^3 - \{\mathbf{0}\}$ such that $\mathbf{C}_\mathrm{T} = \mathscr{H}_{\vec{\alpha}} \cap \{0, ..., n-1\}^2$.

The computation involved in binary search for a border pair is trivial if the search range is a straight line. If the search range is along a zigzag line (e.g., $\mathscr{Z}^+_{p,q}$), we relax the condition for determining the "midpoint" to query at each stage. Assuming the zigzag line has a slope of less than 1 (as is always the case here), we do binary search along the outermost points on the zigzag line in each column. In the case of $\mathscr{Z}^+_{p,q}$ these are the points

$$\left\{\left\langle u, \left\lceil \frac{du}{b}\right\rceil\right\rangle \mid 0 \leqslant u \leqslant b\right\}.$$

The pair $\mathbf{p} \in \mathbf{C}_\mathrm{T}$ and $\mathbf{q} \notin \mathbf{C}_\mathrm{T}$ found is not necessarily adjacent, but there is at most one grid point between them which, when queried, will yield a border pair.

One can represent the affine transformations by 3×3 coefficient matrices. When a new transformation is applied, the current and new transformation matrices are multiplied.

The method for computing a good rational approximation a/b (with a bounded denominator) to a rational number is discussed in [Lov 86]. The basic idea is this. Compute the continued fraction representation of the original rational number. Then find the maximal "prefix" of this continued fraction that represents a rational number with denominator less than the required bound. That rational number a/b will satisfy the bounds stated in Lemma 2.5.

The Extended Euclidean Algorithm (see [AHU74]) can be used to find the value d such that $ad + 1 \equiv 0 \pmod{b}$. Once d is known, the computation of c is trivial.

The equations of a border line $\mathscr{H}_{\vec{\alpha}}$ such that $\mathbf{C}_\mathrm{T} = \mathscr{H}_{\vec{\alpha}} \cap \{0, ..., n-1\}^2$ can be determined by post-processing a list of all $O(\log n)$ points queried by the algorithm. These points can be used to produce a set of $O(\log n)$ inequalities in the coefficients $\vec{\alpha}$ of a line $\alpha_1 x + \alpha_2 y = \alpha_3$ that separates them. There are algorithms for linear programming that will produce a satisfying $\vec{\alpha} \in \mathbb{Q}^3 - \{\mathbf{0}\}$, in time polynomial in the number of points. The computed $\vec{\alpha} \in \mathbb{Q}^3$ can be normalized to a satisfying $\vec{\alpha} \in \mathbb{Z}^3 - \{\mathbf{0}\}$.

This completes the proof of Theorem 2.1. ∎

3. FAST AND EXACT IDENTIFICATION OF POLYGONS WITH MEMBERSHIP QUERIES

It turns out that the number of membership queries that are needed to identify polygons does not depend just on n and the number k of edges; it also depends on the minimum and maximum angle between any two adjacent edges of the polygon. To see this, consider for any $\rho > 0$ the concept class $\mathscr{C}_\rho$ of triangles over $\{0, ..., n-1\}^2$ whose angles are between ρ and $\pi - \rho$ (see Fig. 8). Furthermore, let $\mathscr{D}_\rho$ be the concept class of pentagons and all angles between $\pi/2$ and $\pi - \rho$ (see Fig. 9). One can easily show that $\Omega(1/\rho)$ membership queries are needed for identifying arbitrary target concepts from $\mathscr{C}_\rho$, respectively $\mathscr{D}_\rho$, since single points such as the points $\mathbf{p}$ in Figs. 8 and 9 can only be found by exhaustive search over a set of $\Omega(1/\rho)$ grid points. Therefore, instead of k-HALFSPACE^2_n we consider the following concept class. Assume that $k \in \mathbb{N}^+$ and $0 < \rho \leqslant \pi/2$. Then

k-HALFSPACE$^2_{n,\rho}$
$:= \{\mathbf{C} \subseteq \{0, ..., n-1\}^2 \mid \exists$ halfplanes $\mathscr{H}_1, ..., \mathscr{H}_k \subseteq \mathfrak{R}^2$ such that for $\mathscr{P} := \mathscr{H}_1 \cap \cdots \cap \mathscr{H}_k$ one has $\mathscr{P} \subseteq [1, n-2]^2$ and $\mathbf{C} = \mathscr{P} \cap \{0, ..., n-1\}^2$, and the angle α between any two adjacent edges of the polygon $\mathscr{P}$ satisfies $\rho \leqslant \alpha \leqslant \pi - \rho$, and each edge of the polygon $\mathscr{P}$ has length at least $16 \cdot \lceil 1/\rho\rceil\}$.

The restriction that the polygon is entirely contained within the grid is really insignificant. A general intersection of k halfplanes on an $n \times n$ grid can be construed as an at most $(k+4)$-edged polygon contained within an $(n+2) \times (n+2)$ grid. We take the border lines of the original halfplanes, as well as the sides of the $n \times n$ grid, and add a boundary of new grid points around the $n \times n$ grid.

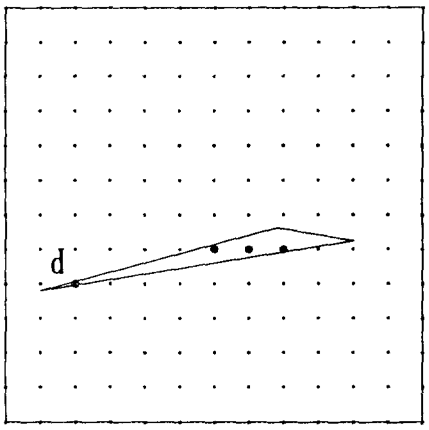


FIG. 8. A concept from the class $\mathscr{C}_\rho$.

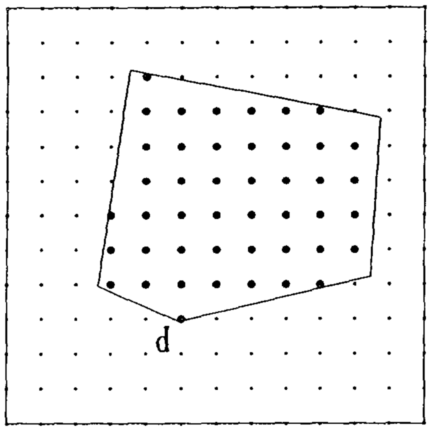


FIG. 9. A concept from the class $\mathscr{D}_p$.

In the case of single halfplanes, we knew that for any (nonempty) concept $C_T \in \text{HALFSPACE}_n^2$ at least one of the four grid corners was in C_T. In the case of polygons (k-HALFSPACE_n^2 or k-$\text{HALFSPACE}_{n,p}^2$), it is no longer easy to find any single point in C_T. In fact, it is easy to give a lower bound of $\Omega((np/k)^2)$ for the required number of membership queries to learn k-$\text{HALFSPACE}_{n,p}^2$ without any given initial point. Hence, we assume in the following that some point $\mathbf{p}_0 \in C_T$ is given initially to the learner.

THEOREM 3.1. *There is a learning algorithm to exactly identify any* $C_T \in k$-$\text{HALFSPACE}_{n,p}^2$ *from any given point* $\mathbf{p}_0 \in C_T$ *with* $O(k(1/p + \log n))$ *membership queries. The number of computation steps of the algorithm is bounded by a polynomial in* k, $\log n$ *and* $1/p$.

Proof. As mentioned above, the polygons we consider are built from halfplanes $\mathscr{H}_{\vec{\alpha}} \subseteq \mathfrak{R}^2$, with $\vec{\alpha} \in \mathbb{Z}^3 - \{\mathbf{0}\}$, where

$$\mathscr{H}_{\vec{\alpha}} := \{\langle x, y\rangle \in \mathfrak{R}^2 \mid \alpha_1 x + \alpha_2 y \geq \alpha_3\}.$$

Associated with any halfplane $\mathscr{H}_{\vec{\alpha}}$ is its *border line* $\mathscr{L}_{\vec{\alpha}}$,

$$\mathscr{L}_{\vec{\alpha}} := \{\langle x, y\rangle \in \mathfrak{R}^2 \mid \alpha_1 x + \alpha_2 y = \alpha_3\}.$$

When three or more halfplanes intersect to form a polygon $\mathscr{P} = \mathscr{H}_1 \cap \cdots \cap \mathscr{H}_k$, the associated border lines $\mathscr{L}_i$ from *edges*,

$$\mathscr{E}_i := \mathscr{L}_i \cap \left(\bigcap_{j \neq i} \mathscr{H}_j\right).$$

We will assume w.l.o.g. that each $\mathscr{E}_i$ has nonzero length, and that they are numbered consecutively, i.e., $\mathscr{E}_i$ intersects only $\mathscr{E}_{i-1}$ and $\mathscr{E}_{i+1}$ (modulo k). The line $\mathscr{L}_i$ is said to be the line extending $\mathscr{E}_i$. The intersection of two edges, or equivalently lines, is a vertex v of $\mathscr{P}$.

Assume some $C_T \in k$-$\text{HALFSPACE}_{n,p}^2$ and $\mathscr{P}_T = \mathscr{H}_1 \cap \cdots \cap \mathscr{H}_k$ such that $C_T = \mathscr{P}_T \cap \{0, ..., n-1\}^2$ have been fixed by the environment. Assume for simplicity that $1/p \in \mathbb{N}$. A rough sketch of the learning algorithm is as follows. From the initial point $\mathbf{p}_0 \in C_T$, we find a small cluster of points in C_T. From this cluster, we search in any direction until we find several neighboring border pairs on some edge of the polygon. We trace roughly along this edge until it ends, and locate the vertex to within a small region. The *exact* trace of the edge between the original border pairs and the vertex region can be learned by the algorithm for $\text{STRIP}_{X,Y}$ from the previous section. The exact trace of the polygon within the small vertex region can be learned by exhaustive search. A few border pairs on the next adjacent edge. After at most $k+1$ repetitions, we will have retraced a known edge and finished. Knowing the trace of all edges and vertices determines C_T by convexity.

The algorithm is presented as a series of subroutines, each corresponding to one sentence in the above paragraph. The flowchart of Fig. 10 gives the subroutine names and their calling sequence. For the remainder of this section, we will present these subroutines in the order indicated by the flowhchart. Interspersed between the subroutines we will describe any nontrivial work that must be done by the main algorithm which calls the subroutines.

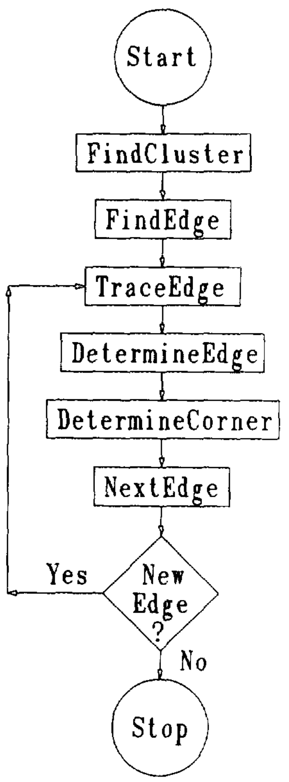


FIG. 10. The flowchart of the learning algorithm for k-$\text{HALFSPACE}_{n,p}^2$.

Along with the written description, a diagram of the action of each subroutine will be presented. This will be represented as a typical input configuration and a possible output configuration resulting from it. In the diagrams, grid points in $\mathbf{C}_T$ will be represented with an $\times$, and grid points not in $\mathbf{C}_T$ will be represented with an $\bigcirc$. The diagrams cannot represent the general case of the computation of the subroutine, but are only intended to ease interpretation of the written description.

Before we begin descriptions of the subroutines, we state a lemma which will be used in many of the subroutines. Informally, it states that at a distance d from the intersection of two lines meeting with angle ρ, the lines will be separated by distance $\Omega(d\rho)$.

LEMMA 3.2. *Let two lines $\mathscr{L}_1$ and $\mathscr{L}_2$ intersect at point v with angle ρ $(0<\rho\leqslant\pi/2)$. Let $\mathscr{S}$ be a length l line segment joining $\mathscr{L}_1$ and $\mathscr{L}_2$. If $\mathscr{S}$ has a distance d from v (as measured by the distance between v and any point on $\mathscr{S}$), then $l\geqslant(2/\pi)\,d\rho$.*

Proof. We prove the lemma for the longest distance d, from which the lemma follows. The longest distance from v to $\mathscr{S}$ will occur along one of the lines $\mathscr{L}_1$ or $\mathscr{L}_2$, say $\mathscr{L}_1$ w.l.o.g. We thus wish to find the minimum value that l can attain for the given ρ and d. Since the shortest distance between a point and a line occurs along a segment perpendicular to the line, we will find the minimum value for l when $\mathscr{S}$ is perpendicular to $\mathscr{L}_2$. The inequality

$$l\geqslant d\sin\rho$$

follows immediately from these observations. Since the quantity $\rho/\sin\rho$ is monotone increasing over $(0,(\pi/2)]$, it follows that

$$\frac{\rho}{\sin\rho}\leqslant\frac{(\pi/2)}{\sin(\pi/2)}$$
$$=\frac{\pi}{2}.$$

Thus,

$$d\rho=(d\sin\rho)\left(\frac{\rho}{\sin\rho}\right)$$
$$\leqslant l\frac{\pi}{2},$$

from which our result follows. This completes the proof of Lemma 3.2. ▌

We now continue with the presentation of the subroutines of the learning algorithm for k-HALFSPACE$^2_{n,\rho}$.

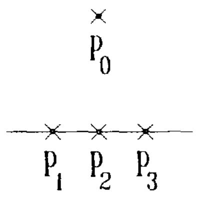


FIG. 11. Diagram of subroutine FINDCLUSTER

FINDCLUSTER $(\mathbf{p}_0\rightarrow\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3)$.

Input: $\mathbf{p}_0\in\mathbf{C}_T$.

Output: Three columnwise (resp. rowwise) adjacent points $\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3\in\mathbf{C}_T$.

Queries: $O(1/\rho)$.

(See Fig. 11.)

Let $\mathbf{S}_d$ be the set of grid points in the square of points at most d units horizontally and vertically from $\mathbf{p}_0=\langle x_0,y_0\rangle$; i.e., for any $d\in\mathbb{N}$, let

$$\mathbf{S}_d:=\{\langle u,v\rangle\in\{0,\ldots,n-1\}^2\mid|v-y_0|\leqslant d \text{ and } |u-x_0|\leqslant d\}.$$

LEMMA 3.3. *There exist three points $\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3$ such that*

(i) $\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3\in\mathbf{C}_T$,

(ii) $\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3\in\mathbf{S}_2\cup(\mathbf{S}_{\lceil 5/\rho\rceil+3}-\mathbf{S}_{\lceil 5/\rho\rceil})$,

(iii) $\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3$ *are columnwise adjacent, or* $\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3$ *are rowwise adjacent.*

Proof. We may assume that no three points in $\mathbf{S}_2$ satisfy (i) and (iii), since otherwise we are done. Thus, some edge $\mathscr{E}_+$ cuts between $\mathbf{p}_0+\langle 0,2\rangle$, and some edge $\mathscr{E}_-$ cuts between $\mathbf{p}_0$ and $\mathbf{p}_0-\langle 0,2\rangle$. Clearly, $\mathscr{E}_+\neq\mathscr{E}_-$. Let $\mathscr{L}_+$ (resp. $\mathscr{L}_-$) be the line extending $\mathscr{E}_+$ (resp. $\mathscr{E}_-$). Then $\mathscr{L}_+$ and $\mathscr{L}_-$ are separated by vertical distance at most 3 at the column of $\mathbf{p}_0$.

Suppose $\mathscr{E}_+$ and $\mathscr{E}_-$ are parallel. Then $\mathscr{L}_+$ and $\mathscr{L}_-$ are separated by vertical distance at most 3 everywhere. Let v be an endpoint of $\mathscr{E}_+$, let $\tilde{\mathscr{E}}$ be the edge adjacent to $\mathscr{E}_+$ at v, and let $\tilde{v}$ be the second endpoint of $\tilde{\mathscr{E}}$. By definition of k-HALFSPACE$^2_{n,\rho}$, $\tilde{\mathscr{E}}$ has length at least $16/\rho$, and meets $\mathscr{E}_+$ at an angle of at least ρ. Applying Lemma 3.2, we see that $\tilde{v}$ must be at least $(2/\pi)\cdot(16/\rho)\cdot\rho=(32/\pi)>10$ units away from $\mathscr{L}_+$. Since $\mathscr{L}_-$ is at most 3 units below $\mathscr{L}_+$, we know that $\tilde{v}$ must be below $\mathscr{L}_-$. This is a contradiction, since all points in $\mathscr{P}_T$ lie on the same side of $\mathscr{L}_-$ as $\mathbf{p}_0$. Thus, $\mathscr{E}_+$ and $\mathscr{E}_-$ are not parallel.

Thus, we may assume that $\mathscr{L}_+$ and $\mathscr{L}_-$ meet on one side of $\mathbf{p}_0$ (say the left side w.l.o.g.). Suppose $\mathscr{E}_+$ and $\mathscr{E}_-$ are not adjacent. Then $\mathscr{L}_+$ and $\mathscr{L}_-$ are separated by a vertical distance at most 3 everywhere left of $\mathbf{p}_0$. The same argument as above, applied on the left side of $\mathbf{p}_0$, gives a contradiction. Thus, $\mathscr{E}_+$ and $\mathscr{E}_-$ are adjacent, with shared endpoint v.

Since $\mathscr{E}_+$ and $\mathscr{E}_-$ must meet at an angle of at least ρ, and they are separated by a vertical distance of at most 3 at the column of $\mathbf{p}_0$, Lemma 3.2 implies that v is at most $3\pi/2\rho < 5/\rho$ units from $\mathbf{p}_0$. Since both $\mathscr{E}_+$ and $\mathscr{E}_-$ have length at least $16/\rho$, they must extend at least $16/\rho - 5/\rho = 11/\rho$ units to the right of $\mathbf{p}_0$. Thus, both $\mathscr{E}_+$ and $\mathscr{E}_-$ pass through $(\mathbf{S}_{(5/\rho)+3} - \mathbf{S}_{(5/\rho)})$ at a distance of at least $5/\rho$ from v. Applying Lemma 3.2 once more show that there will be at least $(2/\pi)\cdot(5/\rho)\cdot\rho = 10/\pi > 3$ columnwise or rowwise adjacent grid points between $\mathscr{E}_+$ and $\mathscr{E}_-$ when they reach the boundary of $\mathbf{S}_{(5/\rho)}$. This completes the proof of Lemma 3.3. ▌

Since $|\mathbf{S}_2 \cup (\mathbf{S}_{(5/\rho)+3} - \mathbf{S}_{(5/\rho)})| = O(\frac{1}{\rho})$, FINDCLUSTER can exhaustively search this region to complete its task.

By assumption, the main algorithm is given a point $\mathbf{p}_0 \in \mathbf{C}_\mathrm{T}$ *initially. After running* FINDCLUSTER, we have three columnwise or rowwise adjacent points in $\mathbf{C}_\mathrm{T}$. If the points are rowwise adjacent, rotate the grid with the affine transformation

$$\mathscr{A}: \langle u, v\rangle \mapsto \langle n-1-v, u\rangle$$

(see Section 2 for a discussion of grid transformations). Thus, we may assume that we have three columnwise adjacent points in $\mathbf{C}_\mathrm{T}$ to give to FINDEDGE.

FINDEDGE $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \rightarrow \mathbf{p}_4, \mathbf{q}_4, \mathbf{p}_5, \mathbf{q}_5, \mathbf{p}_6, \mathbf{q}_6)$.

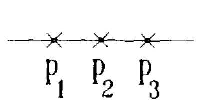
Input: Three columnwise adjacent points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \in \mathbf{C}_\mathrm{T}$.

Output: Three rowwise (resp. columnwise) border pairs $[\mathbf{p}_4, \mathbf{q}_4]$, $[\mathbf{p}_5, \mathbf{q}_5]$, and $[\mathbf{p}_6, \mathbf{q}_6]$ in three adjacent columns (resp. rows) and at most four adjacent rows (resp. columns).

Queries: $O(\log n)$.

(See Fig. 12.)

Let $\mathbf{p}_i = \langle x_i, y_i\rangle$ $(i = 1, 2, 3)$. By definition of k-HALFSPACE$^2_{n,\rho}$, the points on the border of the grid do not belong to $\mathscr{P}_\mathrm{T}$. In particular, the points $\mathbf{q}_i = \langle x_i, 0\rangle$ $(i = 1, 2, 3)$ are not in $\mathbf{C}_\mathrm{T}$. Thus, we can do a binary search between



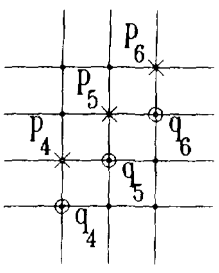


FIG. 12. Diagram of subroutine FINDEDGE.

$\mathbf{p}_i \in \mathbf{C}_\mathrm{T}$ and $\mathbf{q}_i \notin \mathbf{C}_\mathrm{T}$ in each column x_i $(i = 1, 2, 3)$. The result of this binary search is a set of three border pairs $[\mathbf{p}_j, \mathbf{q}_j]$ $(j = 4, 5, 6)$, one in each column searched. If the border pairs occur in four adjacent rows, we are finished.

Suppose the border pairs are separated by at least five rows. Let $[\mathbf{p}_i, \mathbf{q}_i]$ be the lowest and let $[\mathbf{p}_j, \mathbf{q}_j]$ be the highest of these. Let $\mathbf{p}_i = \langle x_i, y_i\rangle$ and $\mathbf{p}_j = \langle x_j, y_j\rangle$. Then it is easy to see that

$$\langle x_i, y_i\rangle, \langle x_i, y_i+1\rangle, \langle x_i, y_i+2\rangle \in \mathbf{C}_\mathrm{T}$$

and

$$\langle x_j, y_i\rangle, \langle x_j, y_i+1\rangle, \langle x_j, y_i+2\rangle \notin \mathbf{C}_\mathrm{T}.$$

Thus, there are border pairs $[\mathbf{p}'_4, \mathbf{q}'_4]$, $[\mathbf{p}'_5, \mathbf{q}'_5]$ and $[\mathbf{p}'_6, \mathbf{q}'_6]$ in rows y_i, (y_i+1) and (y_i+2) respectively. These are columnwise border pairs, and clearly they lie in $3 \leqslant 4$ adjacent columns as well. This satisfies the output requirements of FINDEDGE. As before, if the border pairs returned by FINDEDGE are columnwise, we can make them rowwise by rotating the grid.

In order to proceed with the next step, TRACEEDGE, we need two border pairs that are cut by the same edge of $\mathscr{P}_\mathrm{T}$, and far from a vertex in one direction. Ufortunatly, it is impossible to tell if two arbitrary (even adjacent) border pairs are cut by the same edge. However, there can be at most one vertex between three adjacent border pairs such as those produced by FINDEDGE, since vertices are at least $16/\rho \geqslant 32/\pi > 10$ units apart. Thus, if the leftmost two border pairs either surround a vertex or are less than $8/\rho$ units from a vertex to the left, then the rightmost two border pairs will lie on the same edge and at least $16/\rho - 8/\rho = 8/\rho$ units from the vertex to the right.

The algorithm need not determine which two of the three border pairs lie on the same edge. We run the remainder of *the algorithm on both possibilities, yielding two concepts* $\mathbf{C}_1, \mathbf{C}_2 \in k$-HALFSPACE$^2_{n,\rho}$, one of which is $\mathbf{C}_\mathrm{T}$. If $\mathbf{C}_1 = \mathbf{C}_2$ we are done. Otherwise they differ on some grid point $\mathbf{p} \in \mathbf{C}_1 \triangle \mathbf{C}_2$. By quering this one point $\mathbf{p}$, we can determine which of $\mathbf{C}_1$ or $\mathbf{C}_2$ equals $\mathbf{C}_\mathrm{T}$. It is easy to see that this argument can be generalized to yield a proof of the following claim.

Claim 3.4. Assume for concept classes $\mathscr{C}, \mathscr{C}_1, ..., \mathscr{C}_m$ that one has $\mathscr{C} = \mathscr{C}_1 \cup \cdots \cup \mathscr{C}_m$. Then

$$\mathrm{MEMB}(\mathscr{C}) \leqslant \left(\sum_{i=1}^{m} \mathrm{MEMB}(\mathscr{C}_i)\right) + (m-1).$$

The above claim provides a useful tool to allow a membership query algorithm to make a constant number of "guesses", without increasing its asymptotic complexity. We can now supply two border pairs to TRACEEDGE, which it

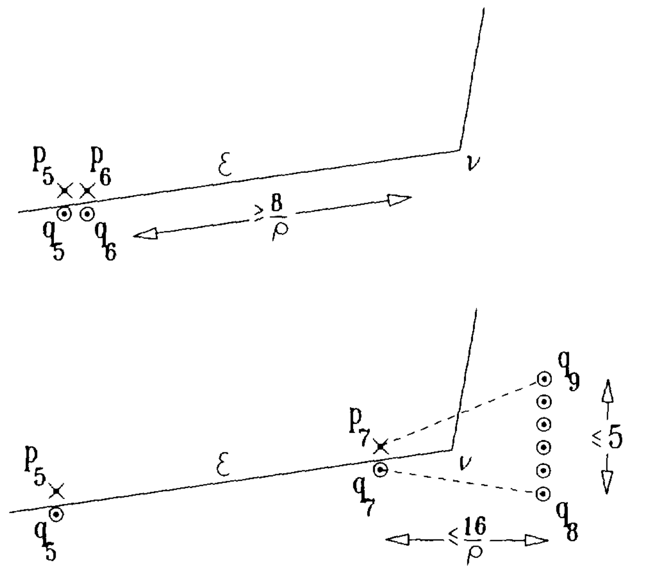


FIG. 13. Diagram of subroutine TraceEdge.

may assume to be on the same edge of $\mathscr{P}_T$ and far from the next vertex.

TraceEdge $(\mathbf{p}_5, \mathbf{q}_5, \mathbf{p}_6, \mathbf{q}_6 \rightarrow \mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9)$

Input: Two rowwise border pairs $[\mathbf{p}_5, \mathbf{q}_5]$, $[\mathbf{p}_6, \mathbf{q}_6]$ in adjacent columns, cut by the same edge $\mathscr{E}$, at least $8/\rho$ units from the endpoint v of $\mathscr{E}$ which is closer to $[\mathbf{p}_6, \mathbf{q}_6]$ than $[\mathbf{p}_5, \mathbf{q}_5]$.

Output: Rowwise border pair $[\mathbf{p}_7, \mathbf{q}_7]$ cut by edge $\mathscr{E}$ and two points $\mathbf{q}_8, \mathbf{q}_9 \notin \mathbf{C}_T$ with $(\mathbf{q}_9 - \mathbf{q}_8) \leqslant 5(\mathbf{p}_7 - \mathbf{q}_7)$, $v \in \operatorname{conv}\{\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9\}$, and the column of $\mathbf{q}_8, \mathbf{q}_9$ is at most $(16/\rho)$ columns from $[\mathbf{p}_7, \mathbf{q}_7]$.

Queries: $O(\log n)$.

(See Fig. 13.)

For the remainder of this section we will refer to conv$\{\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9\}$ as a *vertex region*, since it contains one of the vertices of $\mathscr{P}_T$. We will refer to conv$\{\mathbf{p}_5, \mathbf{q}_5, \mathbf{p}_7, \mathbf{q}_7\}$ as an edge region, since it contains that portion of some edge of $\mathscr{P}_T$ not contained in vertex regions.

TraceEdge is the most complex new part of the k-Halfspace$^2_{n,\rho}$ algorithm. One difficulty is that it never really knows the *exact* location of the edge $\mathscr{E}$ or the vertex v. Furthermore, the discrete set $\mathbf{C}_T$ may be quite complicated. It need not even consist of points from a continuous set of rows (resp. Columns). For example, a "spike" of $\mathscr{P}_T$ may pierce through a column between two adjacent grip points to hit a grid point several columns later (see Fig. 8). Hence, it is a nontrivial task to trace an edge if the domain is the discrete grid.

We begin by performing what we call an inverse binary search to trace the edge $\mathscr{E}$. We know two border pairs $[\mathbf{p}_5, \mathbf{q}_5]$ and $[\mathbf{p}_6, \mathbf{q}_6]$ cut by $\mathscr{E}$ that are d columns apart (initially $d = 1$). Let $\mathscr{L}$ be the line extending $\mathscr{E}$, and $\mathscr{H}$ be the associated halfplane of $\mathscr{P}_T$.

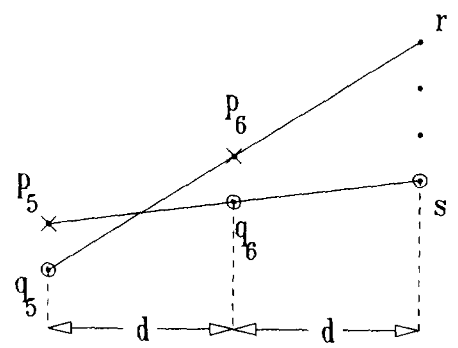


FIG. 14. Inverse binary search along an edge.

Let $\mathbf{r} := 2\mathbf{p}_6 - \mathbf{q}_5$ and $\mathbf{s} := 2\mathbf{q}_6 - \mathbf{p}_5$ (see Fig. 14). The points $\mathbf{r}$ and $\mathbf{s}$ lie in the column $2d$ from $[\mathbf{p}_5, \mathbf{q}_5]$. We know that $\mathbf{s} \notin \mathscr{H}$, since otherwise the point $\mathbf{q}_6 \notin \mathscr{H}$ would lie on a line between the two points $\mathbf{p}_5 \in \mathscr{H}$ and $\mathbf{s} \in \mathscr{H}$, a contradiction to the convexity of halfplanes. Similarly, we know that $\mathbf{r} \in \mathscr{H}$. Assume $\mathscr{E}$ continues to the column of $\mathbf{r}$ and $\mathbf{s}$. Then there is a border pair cut by $\mathscr{E}$ between $\mathbf{r}$ and $\mathbf{s}$. By quering the four adjacent points between $\mathbf{r}$ and $\mathbf{s}$, i.e., those points in

$$\mathbf{S} = \{2\mathbf{p}_6 - \mathbf{q}_5, 2\mathbf{p}_6 - \mathbf{p}_5, 2\mathbf{q}_6 - \mathbf{q}_5, 2\mathbf{q}_6 - \mathbf{p}_5\},$$

we will find this border pair. In fact, we can get by with two queries, since we know that $\mathbf{s} \in \mathscr{H}$, and the others can be found by binary search. We replace $[\mathbf{p}_6, \mathbf{q}_6]$ with this new border pair and repeat the above steps. At each iteration, we double the traced horizontal length of $\mathscr{E}$ with 2 queries. Thus, after at most $2 \log n$ queries, we will have traced beyond the vertex v.

We continue with the inverse binary search until all four points in $\mathbf{S}$ are found to be not in $\mathbf{C}_T$. In particular, this means that the point $\mathbf{r} \notin \mathbf{C}_T$, and so some edge of $\mathscr{P}_T$ must cut between $\mathbf{r}$ and $\mathbf{p}_6 \in \mathbf{C}_T$ (see Fig. 15). Since $\mathbf{r}$ and $\mathbf{p}_6$ lie on the same side of $\mathscr{L}$, it is the next edge $\tilde{\mathscr{E}}$ (which meets $\mathscr{E}$ at v).

As in the proof of Lemma 2.4, we perform a binary search along the zigzag line between $\mathbf{p}_6$ and $\mathbf{r}$ just *outside* of conv$\{\mathbf{p}_6, \mathbf{q}_6, \mathbf{r}, \mathbf{s}\}$. This search reveals a border pair $[\tilde{\mathbf{p}}, \tilde{\mathbf{q}}]$ that is cut by $\tilde{\mathscr{E}}$.

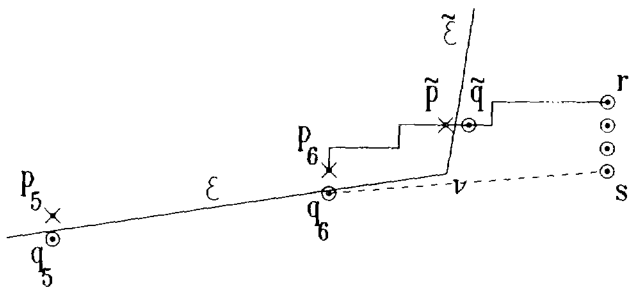


FIG. 15. A new edge cuts between r and $\mathbf{p}_6$.

Let $\mathscr{L}$ (resp. $\tilde{\mathscr{L}}$) be the line extending $\mathscr{E}$ (resp. $\tilde{\mathscr{E}}$). The portion of $\mathscr{L}$ that runs between the columns of $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{q}}$ lies in $\operatorname{conv}\{\mathbf{p}_6, \mathbf{q}_6, \mathbf{r}, \mathbf{s}\}$. Somewhere between those two columns, there are points on $\mathscr{L}$ and $\tilde{\mathscr{L}}$ that are separated by vertical distance at most 5, since $\operatorname{conv}\{\mathbf{p}_6, \mathbf{q}_6, \mathbf{r}, \mathbf{s}\}$ has vertical height at most 3 and zigzag lines have height at most 2. Applying Lemma 3.2, we see that v lies within distance at most $5\pi/2\rho < 8/\rho$ to the left or right of $[\tilde{\mathbf{p}}, \tilde{\mathbf{q}}]$. By the conditions of TRACEEDGE, v is also at least $8/\rho$ units from $[\mathbf{p}_5, \mathbf{q}_5]$.

We set $[\mathbf{p}_7, \mathbf{q}_7]$ to be the border pair in the column $8/\rho$ units from $\tilde{\mathbf{p}}$ in the direction of $[\mathbf{p}_5, \mathbf{q}_5]$, or the column $8/\rho$ units from $[\mathbf{p}_5, \mathbf{q}_5]$ in the direction of $\tilde{\mathbf{p}}$, whichever is closest to $\tilde{\mathbf{p}}$. Thus, edge $\mathscr{E}$ cuts both $[\mathbf{p}_5, \mathbf{q}_5]$ and $[\mathbf{p}_7, \mathbf{q}_7]$. We set $\mathbf{q}_9$ (resp. $\mathbf{q}_8$) to be the point on the line between $\mathbf{q}_5$ and $\mathbf{p}_7$ (resp. $\mathbf{p}_5$ and $\mathbf{q}_7$), in the column $8/\rho$ units from $\tilde{\mathbf{q}}$ in the direction away from $[\mathbf{p}_5, \mathbf{q}_5]$. Thus, line $\mathscr{L}$ also cuts between $\mathbf{q}_8$ and $\mathbf{q}_9$, but only after vertex v. Since the distance between $\mathbf{p}_7$ and $\mathbf{q}_8$ (resp. $\mathbf{q}_7$ and $\mathbf{q}_9$) is at most twice the distance between $\mathbf{p}_7$ and $\mathbf{q}_5$ (resp. $\mathbf{q}_7$ and $\mathbf{p}_5$) we know that the distance between $\mathbf{q}_9$ and $\mathbf{q}_8$ is at most 5 units. We have therefore satisfied the conditions required of TRACEEDGE.

DETERMINEEDGE ($\mathbf{p}_5, \mathbf{q}_5, \mathbf{p}_7, \mathbf{q}_7 \rightarrow \mathbf{E}_+, \mathbf{E}_-$)

Input: Two rowwise border pairs $[\mathbf{p}_5, \mathbf{q}_5]$, $[\mathbf{p}_7, \mathbf{q}_7]$ on the same edge $\mathscr{E}$.

Output: Two sets $\mathbf{E}_+ \subseteq \mathbf{C}_\mathrm{T} \cap \operatorname{conv}\{\mathbf{p}_5, \mathbf{q}_5, \mathbf{p}_7, \mathbf{q}_7\}$, $\mathbf{E}_- \subseteq \overline{\mathbf{C}_\mathrm{T}} \cap \operatorname{conv}\{\mathbf{p}_5, \mathbf{q}_5, \mathbf{p}_7, \mathbf{q}_7\}$ with $|\mathbf{E}_+ \cup \mathbf{E}_-| = O(\log n)$, such that any line separating $\mathbf{E}_+$ from $\mathbf{E}_-$ will correctly separate $\mathbf{C}_\mathrm{T} \cap \operatorname{conv}\{\mathbf{p}_5, \mathbf{q}_5, \mathbf{p}_7, \mathbf{q}_7\}$ from $\overline{\mathbf{C}_\mathrm{T}} \cap \operatorname{conv}\{\mathbf{p}_5, \mathbf{q}_5, \mathbf{p}_7, \mathbf{q}_7\}$.

Queries: $O(\log n)$.

(See Fig. 16.)

It is clear that, after rotating and/or flipping the grid, this is a valid input configuration for the learning algorithm for STRIP$_{X,Y}$ presented in Section 2. We simply run that algorithm as a subroutine, nd keep track of the results of the $O(\log n)$ points queried.

DETERMINEVERTEX ($\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9 \rightarrow \mathbf{V}_+, \mathbf{V}_-$).

Input: Four points $\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9$ with $|\mathbf{p}_7 - \mathbf{q}_7| = \langle 0, 1 \rangle$, $|\mathbf{q}_9 - \mathbf{q}_8| \leqslant \langle 0, 5 \rangle$ and $|\mathbf{q}_8 - \mathbf{q}_7| \leqslant \langle 16/\rho, c \rangle$ for some $c \in \mathbb{N}$.

Output: Two sets $\mathbf{V}_+ = \mathbf{C}_\mathrm{T} \cap \operatorname{conv}\{\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9\}$ and $\mathbf{V}_- = \overline{\mathbf{C}_\mathrm{T}} \cap \operatorname{conv}\{\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9\}$.

Queries: $O(1/\rho)$.

(See Fig. 17.)

This is done by exhaustive search of all grid points within $\operatorname{conv}\{\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9\}$. The number of such points is $O(1/\rho)$.

NEXTEDGE ($\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9 \rightarrow \tilde{\mathbf{p}}_5, \tilde{\mathbf{q}}_5, \tilde{\mathbf{p}}_6, \tilde{\mathbf{q}}_6$).

Input: Rowwise border pair $[\mathbf{p}_7, \mathbf{q}_7]$ cut by edge $\mathscr{E}$ and two points $\mathbf{q}_8, \mathbf{q}_9 \notin \mathbf{C}_\mathrm{T}$ with $(\mathbf{q}_9 - \mathbf{q}_8) \leqslant 5(\mathbf{p}_7 - \mathbf{q}_7)$, with one endpoint v of $\mathscr{E}$ in $\operatorname{conv}\{\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9\}$, where the column of $\mathbf{q}_8, \mathbf{q}_9$ is at most $16/\rho$ columns from $[\mathbf{p}_7, \mathbf{q}_7]$.

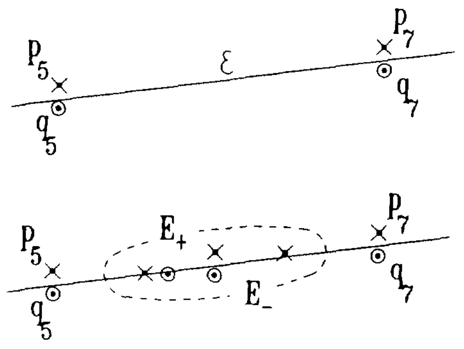


FIG. 16. Diagram of subroutine DETERMINEEDGE.

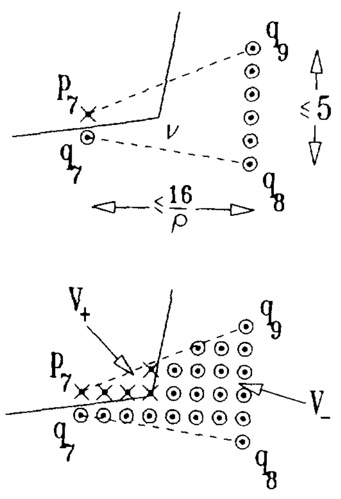


FIG. 17. Diagram of subroutine DETERMINEVERTEX.

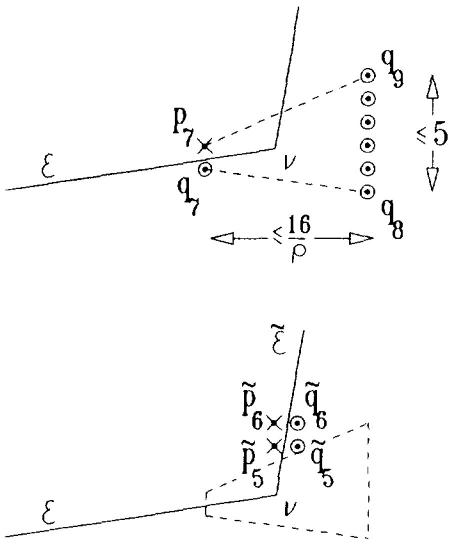


FIG. 18. Diagram of subroutine NEXTEDGE

Output: Two columnwise (resp. rowwise) border pairs $[\tilde{\mathbf{p}}_5, \tilde{\mathbf{q}}_5]$, $[\tilde{\mathbf{p}}_6, \tilde{\mathbf{q}}_6]$ in adjacent rows (resp. columns) both cut by the edge $\tilde{\mathscr{E}}$ adjacent to $\mathscr{E}$ at v.

Queries: $O(1)$.

(See Fig. 18.)

Let $\mathscr{L}$ be the line extending $\mathscr{E}$. Some edge $\tilde{\mathscr{E}}$ of $\mathscr{P}_T$ cuts between $\mathbf{p}_7 \in \mathbf{C}_T$ and $\mathbf{q}_9 \notin \mathbf{C}_T$. Since both $\mathbf{p}_7$ and $\mathbf{q}_9$ lie on the same side of $\mathscr{L}$, $\tilde{\mathscr{E}} \neq \mathscr{E}$. We do a binary search along the zigzag line between $\mathbf{p}_7$ and $\mathbf{q}_9$ just outside $\operatorname{conv}\{\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9\}$. The result of this search is a border pair $[\tilde{\mathbf{p}}, \tilde{\mathbf{q}}]$. In fact, we can use the border pair $[\tilde{\mathbf{p}}, \tilde{\mathbf{q}}]$ found by TRACEEDGE, which is $8/\rho$ columns from $\mathbf{q}_8$ and $\mathbf{q}_9$ by definition. Since $[\tilde{\mathbf{p}}, \tilde{\mathbf{q}}]$ is at most $8/\rho$ columns from $[\mathbf{p}_7, \mathbf{q}_7]$, the edge $\tilde{\mathscr{E}}$ must be adjacent to $\mathscr{E}$ at v. The edge $\tilde{\mathscr{E}}$ must extend at least $16/\rho - 8/\rho = 8/\rho$ units beyond $\operatorname{conv}\{\subseteq \mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9\}$.

We wish to show that within 1 unit of $[\tilde{\mathbf{p}}, \tilde{\mathbf{q}}]$ there will be twio border pairs satisfying the output requirements of NEXTEDGE. Assume that $\tilde{\mathbf{q}} = \tilde{\mathbf{p}} + \langle 0, 1\rangle$ and $[\tilde{\mathbf{p}}, \tilde{\mathbf{q}}]$ lies above $\operatorname{conv}\{\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9\}$; all other cases are handled similarly. Query the points $\tilde{\mathbf{r}} := \tilde{\mathbf{p}} + \langle -1, 1\rangle$ and $\tilde{\mathbf{s}} := \tilde{\mathbf{q}} + \langle 1, 1\rangle$. If $\tilde{\mathbf{r}} \in \mathbf{C}_T$ and $\tilde{\mathbf{s}} \notin \mathbf{C}_T$ then there is a border pair between them. We set $[\tilde{\mathbf{p}}_6, \tilde{\mathbf{q}}_6]$ to be this new border pair, and $[\tilde{\mathbf{p}}_5, \tilde{\mathbf{q}}_5]$ to be $[\tilde{\mathbf{p}}, \tilde{\mathbf{q}}]$. If not, then either $\tilde{\mathbf{r}}, \tilde{\mathbf{s}} \in \mathbf{C}_T$ or $\tilde{\mathbf{r}}, \tilde{\mathbf{s}} \notin \mathbf{C}_T$, since it cannot be that $\tilde{\mathbf{r}} \notin \mathbf{C}_T$ and $\tilde{\mathbf{s}} \in \mathbf{C}_T$.

If $\tilde{\mathbf{r}}, \tilde{\mathbf{s}} \in \mathbf{C}_T$, then we set set $[\tilde{\mathbf{p}}_5, \tilde{\mathbf{q}}_5]$ to be $[\tilde{\mathbf{q}} + \langle 0, 1\rangle, \tilde{\mathbf{q}}]$, and $[\tilde{\mathbf{p}}_6, \tilde{\mathbf{q}}_6]$ to be $[\tilde{\mathbf{s}}, \tilde{\mathbf{s}} - \langle 0, 1\rangle]$. If $\tilde{\mathbf{r}}, \tilde{\mathbf{s}} \notin \mathbf{C}_T$, then we set $[\tilde{\mathbf{p}}_5, \tilde{\mathbf{q}}_5]$ to be $[\tilde{\mathbf{r}} - \langle 0, 1\rangle, \tilde{\mathbf{r}}]$, and $[\tilde{\mathbf{p}}_6, \tilde{\mathbf{q}}_6]$ to be $[\tilde{\mathbf{p}}, \tilde{\mathbf{p}} + \langle 0, 1\rangle]$. Since all of these border pairs are less than $16/\rho$ units from v, they are cut by $\tilde{\mathscr{E}}$. This completes the output requirements for NEXTEDGE.

After the main algorithm finishes with NEXTEDGE, it loops back to TRACEEDGE. In order for TRACEEDGE to be successful, it needs to be given two adjacent border pairs on the same edge, at least $8/\rho$ units from the next vertex in the direction to be traced. We have seen that $[\tilde{\mathbf{p}}_5, \tilde{\mathbf{q}}_5]$ and $[\tilde{\mathbf{p}}_6, \tilde{\mathbf{q}}_6]$ are on the same edge $\tilde{\mathscr{E}}$, and that $\tilde{\mathscr{E}}$ extends at least $8/\rho$ units beyond $\operatorname{conv}\{\mathbf{p}_7, \mathbf{q}_7, \mathbf{q}_8, \mathbf{q}_9\}$.

After at most $k+1$ such iterations, we will have retraced the full length of the first edge, found by FINDEDGE. At that point we will have learned the membership of all points in a continuous, connected region, composed of at most k edge and vertex regions, that entirely contains the edges and vertices of $\mathscr{P}_T$. This completes the learning algorithm for k-HALFSPACE$^2_{n,\rho}$ and the proof of Theorem 3.1. ∎

4. LEARNING BOXES AND OTHER "NATURAL" CLASSES OF POLYGONS

Although k-HALFSPACE$^2_{n,\rho} \subsetneq k$-HALFSPACE^2_n, all "natural" polygons belong to k-HALFSPACE$^2_{n,\rho}$ for some n. Two common operations on a grid of pixels are increasing the resolution and "zooming in" on $\mathbf{C}_T$. By increasing the resolution we mean leaving the geometric target object fixed while decreasing the distance between grid points. By "zooming in" on $\mathbf{C}_T$ we mean scaling up the geometric target object's dimensions while leaving the grid fixed. With either of these operations we increase the number of pixels (grid points) on each edge, while leaving the angles fixed. When each edge finally has at least $16\lceil 1/\rho \rceil$ points on it, the figure can be recognized by the learning algorithm presented in the previous section.

A "natural" class of polygons is the set of polygons whose angles are bounded away from 0 and π by some arbitrary, but fixed, constant ρ. For such polygons, the algorithm of the previous section gives the following result.

COROLLARY 4.1. *Assume that $k \in \mathbb{N}^+$ and $\rho(0 < \rho \leqslant \pi/2)$ are constants. Then any $\mathbf{C}_T \in k$-HALFSPACE$^2_{n,\rho}$ can be identified from some given $\mathbf{p}_0 \in \mathbf{C}_T$ with $O(\log n)$ membership queries and $O(\log^{O(1)} n)$ computation steps. Furthermore, one can augment the learning algorithm so that after $O(\log^{O(1)} n)$ computation steps it outputs the coordinates of the at most k vertices of some polygon $\mathscr{P}$ with $\mathbf{C}_T = \mathscr{P} \cap \{0, ..., n-1\}^2$.*

Proof. The first statement follows immediately from Theorem 3.1. The second statement follows from a reduction to linear programming. By definition of k-HALFSPACE$^2_{n,\rho}$, there is some polygon $\mathscr{P}_T$ such that $\mathbf{C}_T = \mathscr{P}_T \cap \{0, ..., n-1\}^2$. Let v be some vertex of $\mathscr{P}_T$, with incident edges $\mathscr{E}_1$ and $\mathscr{E}_2$. Let $\mathscr{H}_1$ (resp. $\mathscr{H}_2$) be the halfplane associated with $\mathscr{E}_1$ (resp. $\mathscr{E}_2$). Each of the points in the vertex region for v (as in DETERMINEVERTEX) belongs to exactly one of the sets $\mathscr{H}_1 \cap \mathscr{H}_2$, $\mathscr{H}_1 - \mathscr{H}_2$, $\mathscr{H}_2 - \mathscr{H}_1$ or $\overline{\mathscr{H}_1} \cap \overline{\mathscr{H}_2}$. By labeling each of the points in every vertex region with one of the four sets associated with the vertex, we obtain a complete classification of the points in vertex regions with respect to all of the halfplanes $\mathscr{H}_1, ..., \mathscr{H}_k$. Since the total number of points in all vertex regions is $O(k/\rho) = O(1)$, there are at most $4^{O(k/\rho)} = O(1)$ classifications.

For each such classification, we try to find k lines which separate the grid points in vertex regions according to the classification, and are also consistent with the information received from the queries in each edge region. We can do this by solving for each classification k systems of linear inequalities in the unknown coefficients of the separating lines. If each of these systems has a solution for the same classification, we have found a valid polygon $\mathscr{P}$ such that $\mathbf{C}_T = \mathscr{P} \cap \{0, ..., n-1\}^2$. Since the systems will at least all have a solution for the "correct" classification induced by $\mathscr{P}_T$, we will find some valid $\mathscr{P}$ after solving $O(1)$ linear programs. From $\mathscr{P}$ we can easily compute the vertices. ∎

The second extension of Theorem 3.1 deals with the special case of rectangles in general position in the grid. As an immediate corollary of Theorem 3.1 we can recognize, given $\mathbf{p}_0 \in \mathbf{C}_T$, any rectangle whose edges are at least 16 units long, using $O(\log n)$ membership queries. The following theorem improves the minimal required edge length to an optimal 1 unit. Note that, with this improvement, the algorithm below can recognize "lines" as well as boxes. A line appearing on a monitor is typically a rectangle with a width of 1 pixel, centered on the line segment it represents. The proof is intentionally presented in a way that makes extensive use of Claim 3.4, to illustrate the power of that tool.

THEOREM 4.2. *Assume that* $\mathbf{C}_T \in \text{GP-BOX}_n^2$ *has minimal edge length at least* 1, *and that some* $\mathbf{p}_0 \in \mathbf{C}_T$ *is given. Then* $\mathbf{C}_T$ *can be identified with* $O(\log n)$ *membership queries.*

Remark 4.3. We note that both assumptions of Theorem 4.2 are necessary. A 1×1 rectangle can exactly contain any single point in the grid. Thus $\Omega(n^2)$ membership queries are required if no initial point is given. Assume that the point $\langle 2, 2 \rangle$ is in $\mathbf{C}_T$, and one edge length is allowed to be less than 1. Then it is easy to see that $\mathbf{C}_T$ can consist of any of the $(n-4) = \Omega(n)$ concepts $\{\langle 2, 2\rangle, \langle i, 3\rangle\}$ for $2 \leqslant i \leqslant n-3$. Also note that, as in Corollary 4.1 we can compute the vertices of some valid rectangle $\mathscr{R}$ such that $\mathbf{C}_T = \mathscr{R} \cap \{0, ..., n-1\}^2$.

Proof (of Theorem 4.2). We will make use of Claim 3.4 to learn $\mathbf{C}_T$ with $O(\log n)$ membership queries. If both edge lengths are at least 16, then $\mathbf{C}_T \in 4\text{-HALFSPACE}^2_{n,\,\pi/2}$, and we can use the algorithm of Section 3. If both edge lengths are at most 32, then all points in $\mathbf{C}_T$ are within $32\sqrt{2}$ units from $\mathbf{p}_0$. We can exhaustively search all such points with $O(1)$ membership queries, and thus learn $\mathbf{C}_T$. The remaining case, with edge lengts l_1 and l_2 with $1 \leqslant l_1 < 16 < 32 < l_2$, will be handled below. Since all rectangles with minimum edge length at least 1 fall into at least one of these three cases, we are done, by Claim 3.4.

We now consider learning the long, thin rectangles of edge lengths l_1 and l_2 with $1 \leqslant l_1 < 16 < 32 < l_2$. Since both edge lengths are at least 1, every row and every column has at least one point in $\mathbf{C}_T$. Since one edge length is at least 32, there are at least $32/\sqrt{2} > 22$ points in adjacent columns (resp. rows) including $\mathbf{p}_0$ that are in $\mathbf{C}_T$. We can find these points by exhaustive search of $O(1)$ points around $\mathbf{p}_0$. We assume w.l.o.g. that they are in adjacent columns.

Since all points in the bottom row of the grid are not in $\mathbf{C}_T$, we can find a border pair below each of the 22 points, as in FINDEDGE. Since the short edges of the rectangle have length less than 16, at least two of the 22 border pairs must be on the same *long* edge. We again use Claim 3.4 to "guess" which two, as in Section 3. We trace this edge as in TRACEEDGE. When a vertex is roughly located, we immediately retrace the long edge back in the opposite direction. Once both vertices on this edge are roughly known, we apply the learning algorithm for $\text{STRIP}_{X,Y}$ to learn the *exact* portion of the edge between the vertex regions.

The remaining vertices are located within less than 16 units from the two known vertices. Thus, with $O(1)$ membership queries we can learn how $\mathbf{C}_T$ intersection not only the two known vertex regions, but also the two unknown vertex regions and the two short edges. One final use of the learning algorithm for $\text{STRIP}_{X,Y}$ completes the learning on the remaining long edge. This completes the proof of Theorem 4.2. ▮

Remark 4.4. Other learning algorithms for intersections of halfspaces were previously proposed by Baum [Bau90b, Bau90c, Bau90a]). He considers in [Bau90c and Bau90a] a learning model that agrees with the usual PAC-learning model except for the following two features: the learner may in addition make membership queries, and one assumes that the target concept is chosen in a "non-malicious" manner. Baum's learning algorithm works for arbitrary dimensions. If one applies it to the case of two dimensions, it achieves within time $O(\log^{O(1)} n)$ (this is the time which the algorithm of Corollary 4.1 needs for a 100% correct determination of the target concept $\mathbf{C}_T$) with high probability an approximation of $\mathbf{C}_T$ by a hypothesis $\mathbf{H}$ with error $\varepsilon = 1/O(\log^{O(1)} n)$. If one assumes, for example, that the underlying distribution over the domain is the uniform distribution over the grid points $\mathbf{p} \in \{0, ..., n-1\}^2$, this implies that $n^2/O(\log^{O(1)} n)$ grid points may lie in the difference of $\mathbf{H}$ and $\mathbf{C}_T$. The same error bound holds for the learning algorithm of [Bau90b]. This algorithm requires no membership queries, but a substantially larger hypothesis space.

If one focuses only on the number of membership queries that the algorithm of [Bau90c, Bau90a] uses, one can derive from the estimates for the parameter b in [Bau90a] that it needs $\Omega((\log(n^2/E))^2)$ membership queries in order to achieve (with high probability that $|((\mathbf{H}-\mathbf{C}_T) \cup (\mathbf{C}_T - \mathbf{H})) \cap \{0, ..., n-1\}^2| \leqslant E$ (consider the uniform distribution over $\{0, ..., n-1\}^2$). Hence, with $O(\log n)$ membership queries (and substantially more than $O(\log^{O(1)} n)$ random examples) it achieves at best an approximation with error $E = n^{2-(1/\sqrt{\log n})}$.

Another difference between the algorithm of Theorem 3.1 and the algorithm of [Bau90c, Bau90a] is the fact that Baum's algorithm would have to use other points $\mathbf{p} \in \Re^2$ besides the grid points $\mathbf{p} \in \{0, ..., n-1\}^2$ for its membership queries (if one applied it to the learning problem considered here).

ACKNOWLEDGMENTS

We thank Eric Baum and Guörgy Turán for their helpful comments. This paper was written under partial support by NSF-Grant CCR 8903398.

Received June 28, 1991; final manuscript received September 11, 1993

REFERENCES

[AFP90] Angluin, D., Frazier, M., and Pitt, L. (1990), Learning conjunctions of Horn clauses, *in* "The 1990 Symposium on the Foundations of Computer Science," pp. 186–192, IEEE Comput. Soc. Washington, DC.

[AHK89] Angluin, D., Hellerstein, L., and Karpinski, M. (1989), "Learning Read-Once Formulas with Queries," Technical Report TR-89-050, International Computer Science Institute, Berkeley, CA.

[AHU74] Aho, H., Hopcroft, J., and Ullman, J. (1979), "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, MA.

[Ang87a] Angluin, D. (1987), "Learning k-Term DNF Formulas Using Queries and Counterexamples," Technical Report YALEU/DCS/RR-559, Yale University.

[Ang87b] Angluin, D. (1987), Learning regular sets from queries and counterexamples, *Inform. and Control* **75**, 87–106.

[Ang88] Angluin, D. (1988), Queries and concept learning, *Mach. Learning* **2**, 319–342.

[Ang89] Angluin, D. (1989), Equivalence queries and approximate fingerprints, *in* "The 1989 Workshop on Computational Learning Theory," pp. 134–145, Morgan Kaufmann, San Mateo, CA.

[Bau90a] Baum, E. B. (1990), Neutral net algorithms that learn in polynomial time from examples and queries, preprint.

[Bau90b] Baum, E. B. (1990), On learning a union of halfspaces, *J. Complexity* **6**, 67–101.

[Bau90c] Baum, E. B. (1990), Polynomial time algorithms for learning neutral nets, *in* "The 1990 Workshop on Computational Learning Theory," Morgan Kaufmann, San Mateo, CA.

[Hua82] Hua L-K. (1982), "Introduction to Number Theory," Springer-Verlag, New York.

[Lov86] Lovász, L. (1986), "An Algorithmic Theory of Numbers, Graphs, and Convexity," SIAM, Philadelphia.

[MT89] Maass, W., and Turán, G. (1989), On the complexity of learning from counterexamples, *in* "The 1989 Symposium on the Foundations of Computer Science," pp. 262–267, IEEE Comput. Soc., Washington, DC.

[MT90] Maass, W., and Turán, G. (1990), On the complexity of learning from counterexamples and membership queries, *in* "The 1990 Symposium on the Foundations of Computer Science," pp. 203–210, IEEE Comput. Soc. Washington, DC.

[MT91] Maass, W., and Turán, G. (1991), Algorithms and lower bounds for on-line learning of geometrical concepts, *Mach. Learning*, to appear.

[MT94] Maass, W., and Turán, G. (1994), How fast can a threshold gate learn? "Computational learning Theory and Natural Learning Systems: Constraints and Prospects" (S. J. Henson, R. Rivest, and G. Drastal, Eds.), to appear.

[PV88] Pitt, L., and Valiant, L. G. (1988), Computational limitations on learning from examples, *J. Assoc. Comput. Mach.* **35**, 965–984.