

Separability of Point Sets by k -Level Linear Classification Trees

Esther M. Arkin* Delia Garijo† Alberto Márquez† Joseph S. B. Mitchell*
Carlos Seara‡

6th April 2011

Abstract

Let R and B be sets of red and blue points in the plane in general position. We study the problem of computing a k -level binary space partition (BSP) tree to classify/separate R and B , such that the tree defines a linear decision at each internal node and each leaf of the tree corresponds to a (convex) cell of the partition that contains only red or only blue points.

1 Introduction

Consider a set of n points in the plane in general position. Each point is either “red” or “blue”. Let R denote the set of red points and let B denote the set of blue points. We study the separability of R and B by a k -level binary space partition tree. We say that R and B are *separated* by a k -level binary space partition tree, T , if each region in the partition of the plane induced by T is monochromatic (contains only points of R or only points of B). The separating k -level tree T is a recursive partition of the plane into monochromatic and disjoint convex regions using (up to) $2^k - 1$ separating straight cuts (lines, rays or segments). Such a tree T of height k (i.e., with k levels) can be used as a classification tree for red/blue points; we can classify, in time $O(k)$, a new point as “red” or “blue” based on the color associated with the cell (corresponding to a leaf in the tree) in which it is located. See Figure 1.

Related work. The separating k -level tree generalizes simple separability criteria that have been previously studied. The most basic separability criteria for R and B is that of linear separability, which corresponds to a separating 1-level tree: There exists a line separating R and B . Linear separability can be decided in linear time [10]. For sets R and B that are not linearly separable, generalizations include the following separability criteria: A strip (two parallel lines, partitioning the plane into three regions), a wedge (two rays with common

*Applied Mathematics and Statistics, State University of New York, Stony Brook, NY, 11794-3600, USA, {estie, jsbm}@ams.stonybrook.edu

†Departamento de Matemática Aplicada I, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain, {dgarijo, almar}@us.es

‡Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Jordi Girona 1, 08034 Barcelona, Spain, carlos.seara@upc.edu

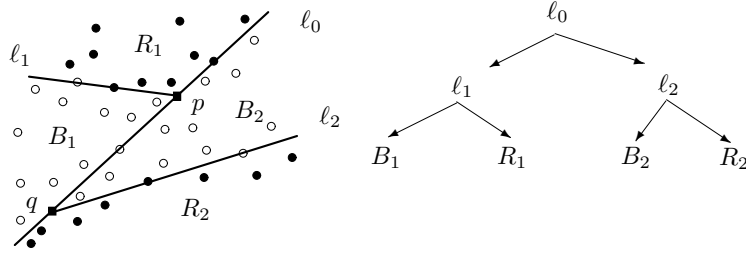


Figure 1: A separating 2-level tree.

origin, partitioning the plane into two regions), a double wedge (two intersecting lines), or three parallel lines. All of these criteria can be decided, and corresponding partitions computed, in optimal $\Theta(n \log n)$ time [1, 2, 8, 9]. (Note that if R and B are strip separable, then they are also wedge separable.) Strip, wedge, double-wedge, or three parallel lines separability criteria are special cases of separability by a 2-level tree.

Separability by multiple parallel lines is a special case of separability by a k -level tree; in particular, $m = 2^k - 1$ parallel lines can be associated with a (height-balanced) k -level tree. The minimum number of parallel lines needed to separate R and B can be computed in $O(n^2 \log n)$ time [2]. If R and B are the vertices of a regular n -gon, $\lfloor n/2 \rfloor$ is a tight upper bound for the number of parallel lines, and, given the minimum number of separating lines, their common orientation can be computed in $O(n \log n)$ time [3].

Other separability criteria have also been studied. Given any disjoint point sets, R and B , there always exists a separating polygonal chain, which can be computed in $O(n \log n)$ time. Computing a minimum-link separating polygonal chain that turns alternatively left and right by a constant angle $\alpha \geq \pi/2$ can be done in $O(n \log n)$ time [8]. Separability by m parallel lines is a special case of separability by a monotone m -link polygonal chain. The problem of determining a minimum-link separating polygonal chain of R and B is NP-complete [6]. Edelsbrunner and Preparata [5] solved, in time $O(n \log n)$, the special case of computing a minimum-edge convex polygon separating R and B (if a convex separator exists); their time bound was shown to be optimal in [1].

Outline of the paper. We initiate the study of separability by k -level trees by considering first the special case of $k = 2$, separability by a 2-level tree. Section 2 is devoted to a special case of 2-level separability, that of separability by a *zigzag*, which corresponds to 2-level tree partitioning such that monochromatic cells of the same color are adjacent (Figure 2). In Section 3 we study the general version of 2-level tree separability, including the generalizations to three or four distinct colors of point sets (instead of just two, red and blue). In Section 4 we consider k -level tree separability and possible configurations of points with $O(\log n)$ -level trees. Section 5 is devoted to separability by k -level trees whose partitioning cuts are axis-parallel.

2 Zigzag Separability

In this section we consider the zigzag separability problem: Determine whether the sets R and B are separable by a *zigzag* $Z = (\ell_1, s, \ell_2)$, which is a simple, nonconvex 3-link

polygonal chain formed by two rays ℓ_1, ℓ_2 and a segment s joining the origins of the rays (Figure 2). Let ℓ_s be the line containing the segment s , and let ℓ'_1 (ℓ'_2) be the line containing the ray ℓ_1 (ℓ_2). Let $CH(X)$ denote the convex hull of a point set X . We can assume that the simpler known special cases of separability have already been tested; specifically, we assume that R and B are not separable by a line, strip, wedge, or convex polygonal chain, each of which can be decided in $O(n \log n)$ time. Thus, under this condition, the following lemma is straightforward.

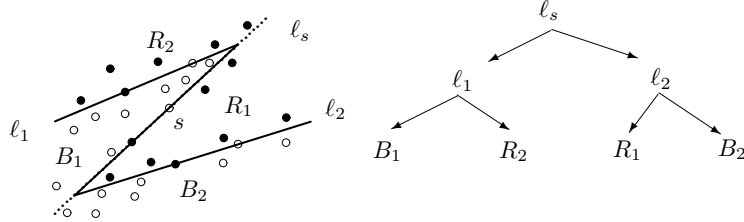


Figure 2: A separating zigzag.

Lemma 1. *Let R and B be zigzag separable but not separable by a line, strip, wedge, or convex polygon. Then, $CH(R)$ contains at least one blue point, and $CH(B)$ contains at least one red point.*

There are three types of zigzags depending on the values of the angles α and β formed by ℓ_s and ℓ_1 , and by ℓ_s and ℓ_2 , respectively (Figure 3). A separating zigzag $Z = (\ell_1, s, \ell_2)$ defines four wedges that partition $R \cup B$ into four subsets, denoted by $R_1, R_2, B_1,$ and B_2 , where $R_2 = R - R_1$ and $B_2 = B - B_1$; all four subsets are non-empty, since R and B are not wedge separable.

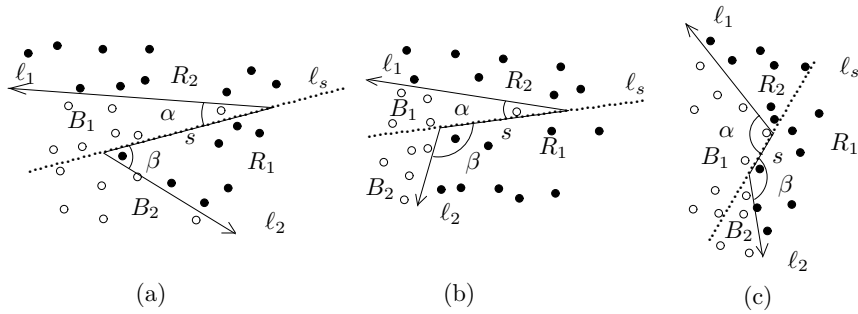


Figure 3: (a) $0 < \alpha, \beta < \pi/2$, (b) $0 < \alpha < \pi/2, \pi/2 \leq \beta < \pi$, (c) $\pi/2 \leq \alpha, \beta < \pi$.

Two optimal separating zigzags are considered: Either a zigzag maximizing $\min\{\alpha, \beta\}$, called the *most convex separating zigzag* (approximating linear separability), or a zigzag that minimizes $\max\{\alpha, \beta\}$ (approximating separability by three parallel lines).

Lemma 2. *Let $Z = (\ell_1, s, \ell_2)$ be the most convex separating zigzag for R and B . Then each of the two rays, and the segment of Z pass through two points of different colors.*

Moreover, either ℓ'_1 is an interior supporting line of $CH(R_2)$ and $CH(B)$, or ℓ'_2 is an interior supporting line of $CH(B_2)$ and $CH(R)$.

Proof. The key idea is to *stretch* the separating zigzag until each part of the structure touches two points of different colors. Moreover, for each of the types of zigzag in Figure 3, either ℓ'_2 intersects ℓ_1 or ℓ'_1 intersects ℓ_2 . In the first case ℓ'_1 is an interior supporting line of $CH(R_2)$ and $CH(B)$, and in the second case ℓ'_2 is an interior supporting line of $CH(B_2)$ and $CH(R)$. Notice that both statements hold if ℓ'_1 and ℓ'_2 are parallel. \square

Lemma 3. *Let R and B be zigzag separable and let $I_{B,R}$ be the number of intersections between pairs of edges of $CH(B)$ and $CH(R)$. Then $I_{B,R} \in \{0, 2, 4, 6\}$.*

Proof. Because the convex hulls are closed Jordan curves, $I_{B,R}$ is even. If $CH(R)$ and $CH(B)$ are nested polygons, $I_{B,R} = 0$ (Figure 4(a)). Assume that $I_{B,R} \geq 2$. By Lemma 2, either $CH(B)$ only intersects $CH(R_1)$, or $CH(R)$ only intersects $CH(B_1)$. Moreover, $CH(R_1)$ and $CH(B)$ are wedge separable; thus, $I_{R_1,B} \leq 4$. Analogously, $I_{B_1,R} \leq 4$ (Figures 4 and 5). Since $CH(R) = CH(R_1 \cup R_2)$, there are two bridge-edges between $CH(R_1)$ and $CH(R_2)$. An analogous statement holds for $CH(B_1)$ and $CH(B_2)$. Hence, $I_{B,R} \leq 4 + 2 = 6$, which corresponds to the at most six alternations of colors in $CH(B \cup R)$ (Figure 5). \square

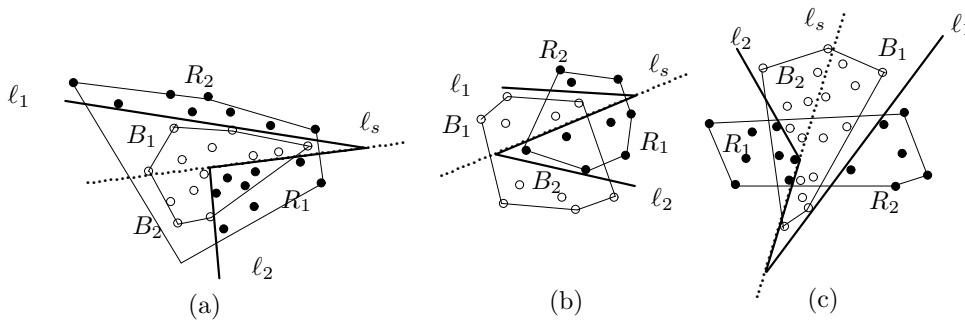


Figure 4: (a) $I_{B,R} = 0$, (b) $I_{B,R} = 2$, and (c) $I_{B,R} = 4$.

Let R_I (B_I) be the subset of red (blue) interior points of $CH(B)$ ($CH(R)$). By Lemma 1, $|R_I| \geq 1$ and $|B_I| \geq 1$. If $I_{B,R} = 6$, let R'_1, R'_2 , and R'_3 (B'_1, B'_2 , and B'_3) be the three disjoint subsets of red (blue) exterior points of $CH(B)$ ($CH(R)$). These eight subsets and their respective convex hulls can be computed in $O(n \log n)$ time (Figure 5).

Lemma 4. *Let $Z = (\ell_1, s, \ell_2)$ be the most convex separating zigzag of R and B . Then ℓ_s is a supporting line of some of the following eight convex polygons: $CH(R_I)$, $CH(B_I)$, $CH(R'_1)$, $CH(R'_2)$, $CH(R'_3)$, $CH(B'_1)$, $CH(B'_2)$, and $CH(B'_3)$.*

Proof. We first prove that either R_I is separable from B by the wedge (ℓ_s, ℓ_2) , or B_I is separable from R by the wedge (ℓ_1, ℓ_s) , and both cases do not always occur (Figure 4 (a) and (c)). By Lemma 2, if R_2 and B are line separable, then R_1 is separable from B by the wedge (ℓ_s, ℓ_2) , and so $R_I \subseteq R_1$ is separable from B by the same wedge. By analogous reasoning, if B_2 and R are line separable, then B_I and R are wedge separable.

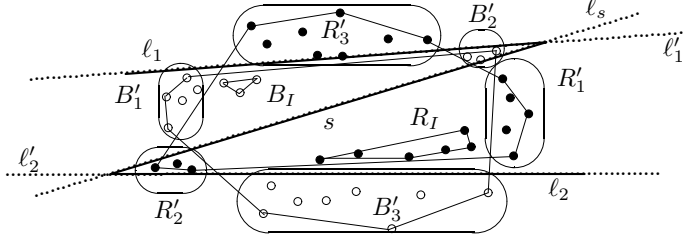


Figure 5: Subsets of red and blue points for $I_{B,R} = 6$.

If $I_{B,R} \in \{0, 2, 4\}$, then ℓ_s is a supporting line of $CH(R_I)$, because, otherwise, there are not red points inside $CH(B)$ and then B is wedge separable from R , since Z is the most convex separating zigzag. Analogously, if B_2 and R are line separable and $I_{B,R} \in \{0, 2, 4\}$, then ℓ_s is a supporting line of $CH(B_I)$.

Assume that $I_{B,R} = 6$, R_2 is line separable from B , and ℓ_s is not a supporting line of $CH(R_I)$. One of the subsets R'_1, R'_2, R'_3 has to be R_2 (say $R'_3 = R_2$) because, by convexity of $CH(B)$, ℓ'_1 does not separate two of these subsets from $CH(B)$. Thus, R'_1 and R'_2 are contained in R_I and, since ℓ_s is not a supporting line of $CH(R_I)$, then ℓ_s is a supporting line of either $CH(R'_1)$ or $CH(R'_2)$ (Figure 5). We can proceed analogously, if B_2 and R are line separable, $I_{B,R} = 6$, and ℓ_s is not a supporting line of $CH(B_I)$. \square

Lemma 4 provides the key tool to design the following $O(n \log n)$ time algorithm for computing a separating zigzag $Z = (\ell_1, s, \ell_2)$ for R and B (if it exists). The algorithm looks for ℓ_s and checks the linear separability of $CH(R_2)$ and $CH(B_1)$ by ℓ'_1 and the linear separability of $CH(R_1)$ and $CH(B_2)$ by ℓ'_2 . There are a linear number of candidates ℓ_s that are supporting lines of the eight convex polygons above.

ZIGZAG-ALGORITHM

Input: R and B

Output: a separating zigzag $Z = (\ell_1, s, \ell_2)$, or report that none exists

1. Compute $CH(R)$, $CH(B)$, R_I , B_I , $CH(R_I)$, $CH(B_I)$, and $I_{B,R}$. Check whether $I_{B,R} \in \{0, 2, 4, 6\}$, and compute the intersecting edges of $CH(R)$ and $CH(B)$. Check that $CH(R_I)$ or $CH(B_I)$ is monochromatic. For $R_I = \{r\}$ and $B_I = \{b\}$, do as follows: If $r \in CH(R)$ and $b \in CH(B)$, then R and B are zigzag separable and it is easy to see how to compute the separating zigzag. Analogously if $r \in CH(R)$ and b is interior to $CH(B)$ or vice versa. From now on assume that $|R_I| \geq 2$ or $|B_I| \geq 2$.
2. Let P be any of the polygons: $CH(R_I)$, $CH(B_I)$, $CH(R'_1)$, $CH(R'_2)$, $CH(R'_3)$, $CH(B'_1)$, $CH(B'_2)$, or $CH(B'_3)$, with their interior points. Do the following:
 - (a) Sort the points in $(R \cup B) - P$ by a counterclockwise rotational sweep over P with an oriented supporting line ℓ_s according to Lemma 4.
 - (b) Do a second rotational sweep over P . Each time ℓ_s bumps a red or blue point of $(R \cup B) - P$, maintain and update the convex hulls $CH(R_2)$, $CH(B_1)$ ($CH(R_1)$, $CH(B_2)$) of the red and blue points on the left (right) side of ℓ_s in $O(\log n)$ time [11]. In $O(\log n)$ time, check the linear separability between $CH(R_2)$ and

$CH(B_1)$, and between $CH(R_1)$ and $CH(B_2)$, and compute their respective supporting lines (Figure 6). In the affirmative case, a separating zigzag is found.

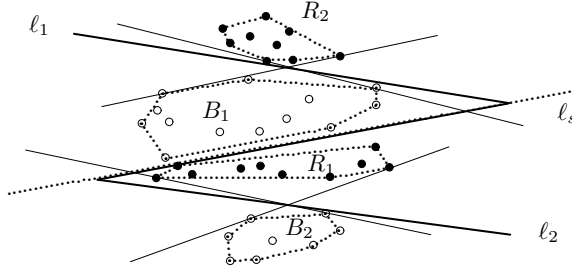


Figure 6: Supporting lines between monochromatic convex hulls.

Analysis of the algorithm. Each step can be done in $O(n \log n)$ time. In step 2 a rotational sweep is done over eight different convex polygons, spending $O(n \log n)$ time on each.

To prove the $\Omega(n \log n)$ time lower bound for deciding the zigzag separability, we reduce the strip separability problem [1] to the zigzag separability problem. The reader is referred to [1] for the construction of the reduction. Put red and blue points on two concentric circles with adequate radii. A small modification is needed: We put blue points in the smallest circle and red points in the biggest circle, plus two red points r_1 and r_2 as in Figure 7. We also put two blue points b_1 and b_2 faraway enough of the biggest circle in adequate positions (Figure 7). It is easy to see that there exists a separating zigzag of the sets of red and blue points if and only if the same sets of red and blue points without b_1 and b_2 are strip separable. The last statement is reduced to determining whether there exist two consecutive blue points in the first quadrant of the smallest circle, such that their Euclidean distance is greater than a given $\epsilon > 0$, which is in the input of the problem.

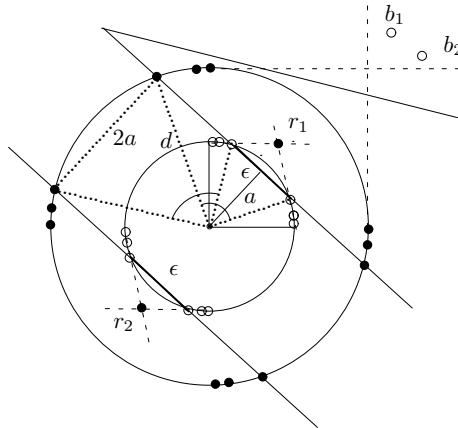


Figure 7: Construction for the lower bound for zigzag separability.

Theorem 1. *Computing a separating zigzag for R and B requires $\Theta(n \log n)$ time.*

Remark. An $O(n^3 \log n)$ time algorithm for determining the separability of R and B by a *monotone* ($k \leq 7$)-polygonal chain is as follows: A mid-segment of the polygonal chain is defined by a line ℓ going through two points. Then, we apply an $O(n \log n)$ time algorithm for the line, wedge or zigzag separability of the point subsets on both sides of ℓ .

3 Separability by a 2-Level Tree

We turn now to the problem of computing a separating 2-level tree $T = (\ell_1, \ell_0, \ell_2)$ for R and B , where ℓ_0 , ℓ_1 , and ℓ_2 are the oriented line, the ray on the left side of ℓ_0 , and the ray on the right side of ℓ_0 , respectively (recall Figure 1). Let ℓ'_1 (ℓ'_2) be the line containing ℓ_1 (ℓ_2). Denote by $m(\ell)$ the slope of ℓ . Let p (q) be the intersection point of ℓ_0 and ℓ_1 (ℓ_2). T splits the plane into four convex regions. Recall that R and B are separated by a 2-level tree if there exists a partition of $R \cup B$ into four monochromatic subsets and a 2-level tree, T , whose partition of the plane respects the partition of $R \cup B$.

Criteria. The following criteria provide a systematic classification of possible separating 2-level trees: (1) $m(\ell_0) > 0$, $m(\ell_0) < 0$, or ℓ_0 is horizontal or vertical. (2) Relative position of p and q along ℓ_0 : $p \preceq q$ or $q \preceq p$. (3) Slopes of ℓ_1 and ℓ_2 with respect to ℓ_0 . (4) Different color assignments to the convex regions.

Classification. We reduce to the following cases: (1) Slope of ℓ_0 : We only consider the $m(\ell_0) \geq 0$ case. The configuration of points where $m(\ell_0) < 0$ can be analyzed by rotating this configuration by 90 degrees and applying the $m(\ell_0) > 0$ case (Figure 8); the case in which ℓ_0 is vertical is symmetric to the ℓ_0 horizontal case, by a 90-degree rotation. (2) Relative position of p and q : We only study the case $q \preceq p$. By applying symmetry with respect to a vertical line, followed by a 90-degree rotation, we get the case $p \preceq q$ (Figure 8). (3) If two consecutive regions have the same color, it corresponds to some of the following criteria: Linear, zigzag ($p \neq q$), or wedge separability ($p = q$) which can be solved in $\Theta(n \log n)$ time [1, 8, 9]. Thus, we assume that the colors alternate, ℓ_0 has positive slope or is horizontal, and $q \preceq p$.

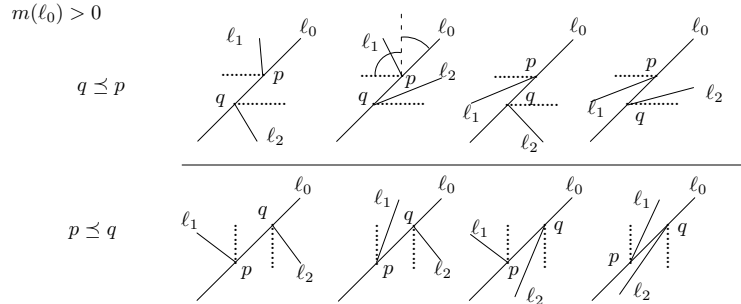


Figure 8: $m(\ell_0) > 0$.

For an easier analysis of the point configurations, we refine the four cases in Figure 8 for $q \preceq p$ into the seven cases in Figure 9 which can be reduced as follows: Case (d) is obtained from case (b) by a 180-degree rotation; case (e) is obtained from case (c) by a 180-degree rotation; and case (g), where ℓ'_2 intersect ℓ_1 , is obtained from case (f), where ℓ'_1

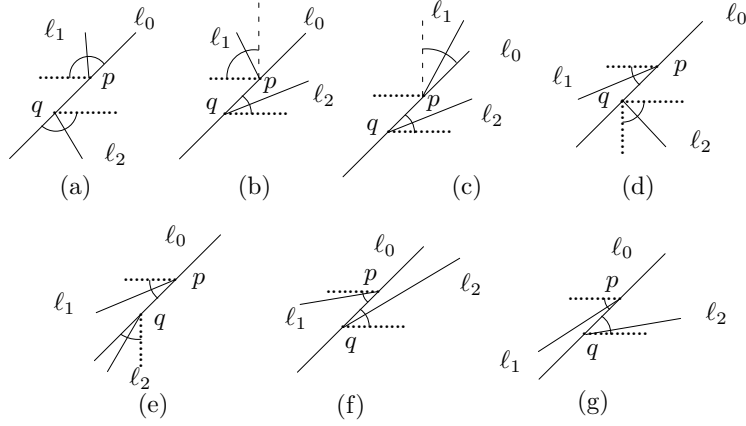


Figure 9: Configurations for $m(\ell_0) > 0$ and $q \preceq p$.

intersect ℓ_2 , by a 180-degree rotation. Thus, we only consider the four types (1), (2), (3), and (4) of 2-level trees in Figure 10 with a concrete assignment of colors. For types (2), (3), and (4), the line ℓ'_1 always intersects ℓ_2 .

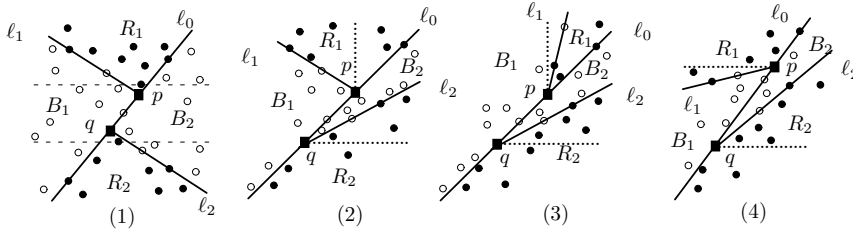


Figure 10: The 4 types of 2-level trees up to symmetry.

We design algorithms for the types of 2-level trees $T = (\ell_1, \ell_0, \ell_2)$ illustrated in Figure 10. From now on, we assume that R and B are not separable by a line, wedge, strip, zigzag, or convex polygonal chain. The following lemma is straightforward.

Lemma 5. *If R and B are separable by a 2-level tree, then $I_{B,R} \in \{0, 2, 4, 6\}$, where $I_{B,R}$ is the number of intersections between pairs of edges of $CH(B)$ and $CH(R)$.*

Lemma 6. *If R and B are separable by a 2-level tree $T = (\ell_0, \ell_1, \ell_2)$, then it holds that: (i) ℓ_0 is a supporting line of $CH(R_1)$ or $CH(R_2)$, and (ii) ℓ'_1 (ℓ'_2) is a common supporting line of $CH(R_1)$ and $CH(B_1)$ ($CH(R_2)$ and $CH(B_2)$).*

Proof. Take any 2-level tree in Figure 10. Turn counterclockwise ℓ_0 with pivot in p until it bumps into a point r from $CH(R_1)$ or $CH(R_2)$, say $r \in CH(R_1)$. Then turn counterclockwise ℓ_0 with pivot in r (or consecutive points in $CH(R_1)$) until it bumps into either a point of $CH(R_2)$ or a blue point that will pass to the convex region containing R_1 if we continue turning. This process can modify the partition B_1, B_2 but still existing a separating 2-level tree. The lines ℓ'_1 and ℓ'_2 are turned in a similar way until they become supporting lines. \square

3.1 Algorithms

An overview of the algorithm is as follows: Compute a line that classifies/separates one of the point sets (say R) into subsets R_1 and R_2 , and use this classification to look for a classification of B into subsets B_1 and B_2 according to a 2-level tree. We present an optimal $O(n \log n)$ time algorithm for 2-level trees of type (1), and we show an $O(n^2)$ time algorithm for 2-level trees of types (2), (3), and (4).

3.1.1 Type (1)

If there exists a 2-level tree of type (1), then there exists a horizontal line between two consecutive red points, r_i and r_{i+1} , in the y -coordinate order, which classifies correctly R into R_1 and R_2 , according to the classification by a separating 2-level tree. We assume that there are no red points with the same y -coordinate. Let h_i be the horizontal line through r_i . Let uR_i (dR_i) be the set of red points that are above (below) or on h_i . Let luB_i and ruB_i (ldB_i and rdB_i) be the subsets of blue points that are above (below) or on h_i and to the left or right of the vertical line going through r_i , respectively. Let mB_i be the set of blue points between h_i and h_{i+1} . Let $a_i = |mB_i|$, with $a_1 + \dots + a_{n-1} = O(n)$ (Figure 11).

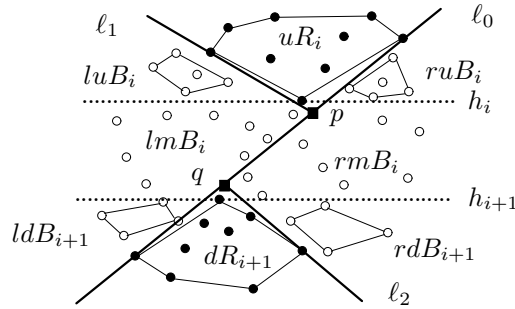


Figure 11: Subsets in a 2-level tree of type (1).

TYPE (1)-ALGORITHM

Input: R and B

Output: a separating 2-level tree $T = (\ell_1, \ell_0, \ell_2)$ of type (1), or report that none exists

1. In $O(n \log n)$ time compute the sequences (r_1, r_2, \dots, r_n) and (b_1, b_2, \dots, b_n) of the red and blue points, respectively, sorted by decreasing y -coordinate.
2. For $i = 1$ to $n - 1$ proceed as follows:
 - (a) Compute $uR_i, dR_{i+1}, luB_i, ruB_i, ldB_{i+1}, rdB_{i+1}, mB_i$. Their respective convex hulls can be computed and updated in amortized $O(n \log n)$ time: At each step we update two red points and a_i blue points. At the end of the process, we have updated seven convex hulls with total $O(n)$ updates (insertions and deletions), where each update is done in $O(\log n)$ time.

The classification of blue points of luB_i and ruB_i is maintained from h_i to h_{i+1} and these blue points can not be inside $CH(uR_i)$, which is verified in $O(\log n)$

time by checking whether $CH(luB_i)$ and $CH(ruB_i)$ do not intersect $CH(uR_i)$. The new vertical line through r_{i+1} has to maintain this classification until we get the correct h_i (if it exists).

The blue points of ldB_{i+1} and rdB_{i+1} maintain their classification while they do not belong to mB_i . Then they will be re-classified at the step they belong either to luB_i or ruB_i . So, the blue points are classified at most two times.

- (b) Check that: (i) $CH(uR_i)$ is line separable from $CH(luB_i)$ and from $CH(ruB_i)$, (ii) $CH(dR_{i+1})$ is line separable from $CH(ldB_{i+1})$ and from $CH(rdB_{i+1})$, and (iii) $CH(uR_i) \cup CH(luB_i) \cup CH(ldB_{i+1})$ is line separable from $CH(dR_{i+1}) \cup CH(ruB_i) \cup CH(rdB_{i+1})$. These conditions are necessary but not sufficient. Since $m(\ell_0) > 0$, $m(\ell_1) < 0$, and $m(\ell_2) < 0$, it holds that $luB_i \cup ldB_{i+1} \in B_1$ and $ruB_i \cup rdB_{i+1} \in B_2$.
- (c) By Lemma 6, ℓ_0 is a supporting line of $CH(R_1)$, i.e., the current $CH(uR_i)$. (Proceed analogously if ℓ_0 is a supporting line of $CH(R_2)$). Let ℓ be an oriented supporting line of $CH(uR_i)$. In $O(a_i \log n)$ time do the following: (1) Sort the points of mB_i according to a rotating sweep with ℓ over $CH(uR_i)$; (2) do a second rotating sweep: Each time a point of mB_i changes from ℓ^- to ℓ^+ , update and maintain $CH(lmB_i)$ (blue points of mB_i in ℓ^-) and $CH(rmB_i)$ (blue points of mB_i in ℓ^+), and compute a line ℓ_0 (if it exists) separating $CH(uR_i) \cup CH(luB_i) \cup CH(ldB_{i+1}) \cup CH(lmB_i)$ from $CH(dR_{i+1}) \cup CH(ruB_i) \cup CH(rdB_{i+1}) \cup CH(rmB_i)$.
- (d) Compute ℓ_1 and ℓ_2 as follows: (1) ℓ'_1 has to separate $CH(uR_i)$ from $CH(luB_i) \cup CH(lmB_i) \cup CH(ldB_{i+1})$. If they are line separable, compute their interior supporting lines, and the intersections of them with ℓ_0 , getting an interval $[a, b]$ on ℓ_0 with $p \in [a, b]$. (2) ℓ'_2 has to separate $CH(dR_{i+1})$ from $CH(ruB_i) \cup CH(rmB_i) \cup CH(rdB_{i+1})$. Proceeding analogously, compute an interval $[c, d]$ on ℓ_0 with $q \in [c, d]$. If $[a, b] \cap [c, d] \neq \emptyset$, then take as $p = q$ any point in $[a, b] \cap [c, d]$, otherwise $p \neq q$.

3. Construct the separating 2-level tree T with the above information in constant time. The monochromatic subsets of the separating 2-level tree are: $R_1 = uR_i$, $R_2 = dR_{i+1}$, $B_1 = luB_i \cup ldB_{i+1} \cup lmB_i$, and $B_2 = ruB_i \cup rdB_{i+1} \cup rmB_i$.

Analysis of the algorithm. Notice that $CH(lmB_i)$ and $CH(rmB_i)$ can be computed in $O(n \log n)$ amortized time because $mB_i \cap mB_{i+1} = \emptyset$ and $O(a_1 \log n + \dots + a_{n-1} \log n) = O(n \log n)$. To maintain the convex hulls of ldB_{i+1} and rdB_{i+1} we can do a horizontal sweep of B from bottom to top. Thus, steps 1 and 2 can be done in $O(n \log n)$ time and the total time complexity of the algorithm is $O(n \log n)$.

3.1.2 Types (2), (3) and (4)

Let (b_1, \dots, b_n) be the sequence of blue points sorted by increasing x -coordinate. Let f_1 and f_2 be vertical lines through b_1 and b_n respectively. Let R''_1 (R''_2) be the set of red points of R_1 (R_2) between f_1 and f_2 . Since R and B are not wedge or strip separable, then $B_1 \neq \emptyset$, $B_2 \neq \emptyset$, and either $R''_1 \neq \emptyset$ or $R''_2 \neq \emptyset$ (Figure 12). Since the line ℓ'_1 always

intersects ℓ_2 , we assume that there is at least a blue point of B_2 inside the wedge (ℓ_0, p, ℓ_1) because otherwise R and B are zigzag separable.

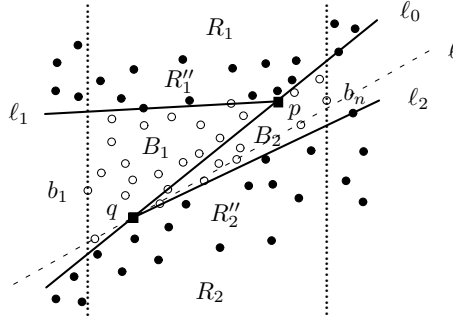


Figure 12: First and last blue points, and line ℓ separating R_1 and R_2 .

Lemma 7. *If R and B are separable by a 2-level tree, then one of the following cases holds: (i) $b_1 \in B_1, b_n \in B_2$; (ii) $b_1 \in B_2, b_n \in B_1$; (iii) $b_1, b_n \in B_2$; (iv) $b_1, b_n \in B_1$.*

Lemma 8. *If R and B are separable by a 2-level tree T , then for any point $b \in B_2$ there is a line ℓ through b that separates R into R_1 and R_2 according to the separation by T .*

Proof. For any separating 2-level tree $T = (\ell_1, \ell_0, \ell_2)$ in Figure 10 do the following: Take a line ℓ through q and rotate ℓ from ℓ_2 to ℓ_0 . During the rotation ℓ passes through all of the points in B_2 and separates R into R_1 and R_2 according to the separation by T . \square

We provide an algorithm for computing a 2-level tree T of type (2), (3), or (4) for R and B based on Lemmas 5, 6, 7, and 8. First we show how to compute a point $b \in B_2$ in order to apply Lemma 8 to get the classification of R into R_1 and R_2 produced by T . By Lemma 7 we get a point $b \in B_2$ if either $b_1 \in B_2$ or $b_n \in B_2$.

Let $b_1, b_n \in B_1$. If T is type (2) or (4), R and B are zigzag separable (Figure 13(a)). If T is type (3) we compute a point in B_2 as follows: Consider a configuration of red and blue points as in Figure 13(b). Sort the red points by increasing x -coordinate and make a vertical red ray from each red point pointing to the south. There is at least a point $r \in R_1$ inside $CH(B)$, otherwise R and B are zigzag separable. The red ray from r intersects an edge $e = b'b \in CH(B)$ with an extreme in B_2 because it does not intersect $CH(B_1)$. We take e as the first edge of $CH(B)$ (counterclockwise order) intersected by a red ray and then either b' or b is in B_2 . The extremes b' and b can be computed in $O(n \log n)$ time.

TYPES-(2)-(3)-(4)-ALGORITHM

Input: R and B

Output: all of the separating 2-level trees T of types (2), (3) or (4) for R and B

1. In $O(n^2)$ time construct the dual arrangement \mathcal{A} of lines from the points in $R \cup B$.
2. In $O(n \log n)$ time do the following: (i) Check that $I_{B,R} \in \{0, 2, 4, 6\}$; (ii) R and B are not wedge, strip, or zigzag separable; and (iii) compute a point $b \in B_2$ according to Lemma 7 and the discussion above. Let ℓ be a directed line through b . Sort the red points according to a rotational sweep with ℓ .

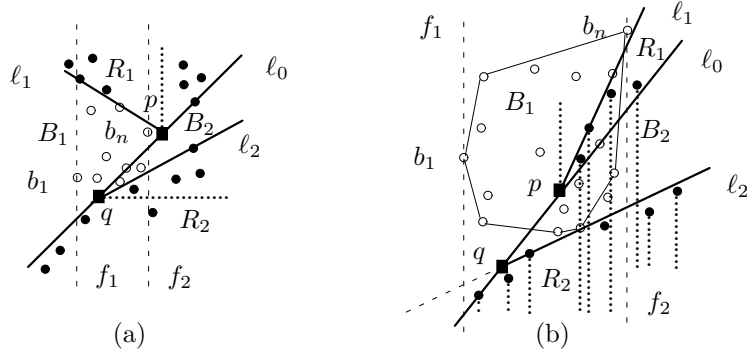


Figure 13: $b_1, b_n \in B_1$.

3. Do a rotational sweep with ℓ stopping each time ℓ bumps a red point getting $O(n)$ partitions $\{R_1, R_2\}$ of R , maintaining $CH(R_1)$, $CH(R_2)$, and the directed supporting line ℓ_R between them such that $CH(R_1)$ is on ℓ_R^- (the left half-plane) and $CH(R_2)$ is on ℓ_R^+ (the right half-plane). For each $\{R_1, R_2\}$ determine whether there exists a bipartition $\{B_1, B_2\}$ of B corresponding to a separating 2-level tree T for R and B in $O(n)$ time as follows:
 - (a) In $O(n)$ time, read in \mathcal{A} the clockwise order of the blue points with respect to $CH(R_1)$ and with respect to $CH(R_2)$, and also check that $CH(R_1)$ and $CH(R_2)$ contain no blue points. Noting that all of the blue points in ℓ_R^+ have to belong to B_2 , let b'_1 be the first blue point in ℓ_R^+ according to the clockwise rotation order. Denote by $(b'_1, b'_2, \dots, b'_n)$ such ordering (Figure 14). It holds that b'_1 belongs to B_2 and b'_n has to belong B_1 . Let ℓ_n be the directed supporting line of $CH(R_1)$ through b'_n , and let ℓ_n^- (ℓ_n^+) be the left (right) half-plane defined by ℓ_n . Obviously, R_1 is contained in ℓ_n^+ .
 - (b) Starting at b'_n and following the order above, in $O(n)$ time compute the location of the blue points in ℓ_n^+ or ℓ_n^- . Let b'_i be the last blue point in $\ell_n^+ \cap \ell_R^-$. If there exists a right partition $\{B_1, B_2\}$, then $\{b'_i, b'_{i-1}, \dots, b'_1\} \subseteq B_2$.
 - (c) Let $B_1 = \{b'_n, \dots, b'_{i+1}\}$ and $B_2 = \{b'_i, b'_{i-1}, \dots, b'_1\}$. By construction, R_1 and B_1 are line separable by ℓ_n . In $O(n)$ time, check if R_2 and B_2 are line separable, and if $R_1 \cup B_1$ is line separable from $R_2 \cup B_2$. Otherwise, it does not exist a separating 2-level tree for the bipartition $\{R_1, R_2\}$. In the affirmative case, construct a separating 2-level tree T for R and B .

Theorem 2. *Computing all of the separating 2-level trees for R and B can be done in $O(n^2)$ time and space.*

Proof. The algorithm above spends $O(n^2)$ time for constructing the dual arrangement \mathcal{A} . For each partition $\{R_1, R_2\}$ of R , the algorithm decides whether there exists a separating 2-level tree and computes it in $O(n)$ time. Lemma 8 ensures the existence of the correct bipartition of R at some step if there exists a separating 2-level tree for R and B . By Lemma 6 we can assume that ℓ is a supporting line of $CH(R_1)$ and proceed analogously

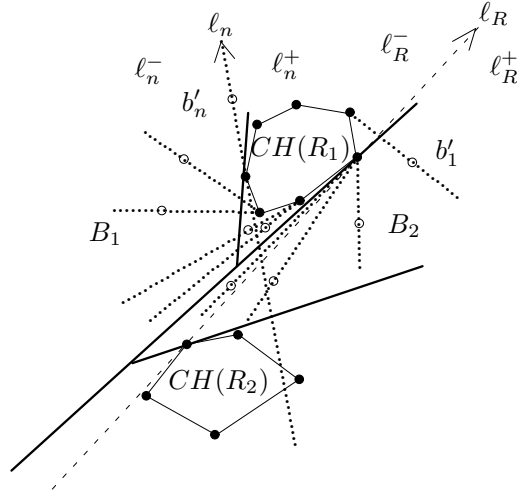


Figure 14: Illustrating the step 3 of the algorithm for a 2-level tree of type (2).

for ℓ being a supporting line of $CH(R_2)$. By the same lemma we can assume that ℓ_n is supporting line of $CH(R_1)$ and $CH(B_1)$. \square

Remark. It is easy to see that all of the combinatorially different 2-level trees for a set of n red and blue points in \mathbb{R}^d can be computed in $O(n^{d+1})$ time. For $d = 2$, the above theorem shows that an improved time bound of $O(n^2)$ is possible. It remains an open problem to determine if a separating 2-level tree for R and B can be computed in $o(n^2)$ time.

3.2 Three or four colored point sets

The 2-level tree problem for three or four colored point sets can be solved as follows. The four colors case can be solved in $O(n)$ time by checking the linear separability between pairs of point sets. The case with three colors, say R , B and G , can be viewed either as a zigzag of R and $G \cup B$, or as a separation with a 2-level tree of R and $G \cup B$ restricted to the linear separability between G and B (Figure 15). But in both cases, we have as additional information the linear separability of G and B , which can be checked previously in $O(n)$ time. We use this information to compute the corresponding 2-level tree. Thus, this problem can be solved in $\Theta(n \log n)$ time. The optimality comes from the zigzag separability for R , B and G easily adapting the lower bound construction in Theorem 1.

Theorem 3. *A separating 2-level tree for three colored sets of n points can be computed in $\Theta(n \log n)$ time. For four colored sets of n points a 2-level tree can be computed in $O(n)$ time.*

4 k -Level Trees

We now consider separating ($k \geq 3$)-level trees for R and B . A separating $O(\log n)$ -level tree for R and B can be computed as follows: Appealing to the Ham-Sandwich theorem,

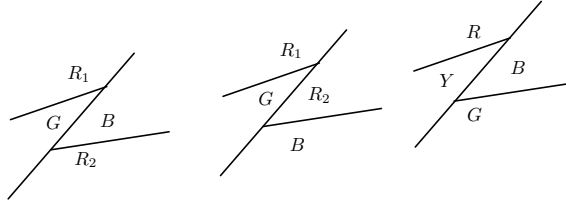


Figure 15: 2-level trees for three and four colored point sets.

we can compute a line that gives an equitable bipartition $B_1 \cup R_1, B_2 \cup R_2$ of $B \cup R$, then proceed recursively on each part until we obtain monochromatic subsets. In the end, we obtain a k -level tree for $n \leq 2^k$ points.

Note that a k -level tree produces a subdivision of the plane into monochromatic convex cells, each one bounded by at most k lines. We can use a dynamic programming algorithm to compute a minimum-level tree for R and B in (quasi-polynomial) $n^{O(\log n)}$ time. In particular, a subproblem is specified by a convex polygon P having at most $k = O(\log n)$ sides, each defined by one of the $\binom{n}{2}$ lines determined by point pairs of $R \cup B$. The optimization for a subproblem selects among the $\leq \binom{n}{2}$ possible cuts, ℓ , and recursively solves the minimum-level tree problem on each side of ℓ .

Theorem 4. *A separating k -level tree for R and B exists with $k \leq \lceil \log n \rceil$. Furthermore, a minimum-level tree can be computed in $n^{O(\log n)}$ time.*

On the other hand, there exist point configurations for which the depth k^* of a minimum-level tree is $\Omega(\log n)$. In particular, let S be a set of n points in general position. Replace each point $p_i \in S$ by a structure of four points, two red and two blue, as in Figure 16, obtaining the sets R and B of $2n$ red and $2n$ blue points, respectively. Any k -level tree for R and B has to separate each pair of red and blue points of the p_i structure. The nodes of the penultimate level of the tree can be the $2n$ half-lines separating these pairs of points such that the leaves of the k -level tree are the $4n$ points of R and B .

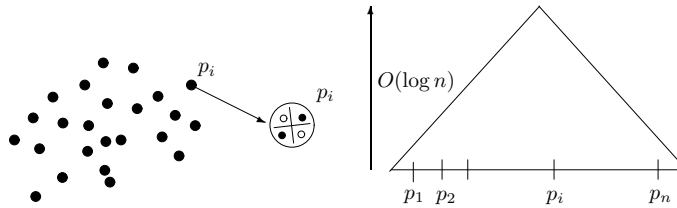


Figure 16: An example (left) of a configuration of $R \cup B$ requiring an $\Omega(\log n)$ -level separating tree.

Grigni et al. [7] considered the problem of designing a near-optimal linear decision tree T to classify two given point sets R and B in \mathbb{R}^n , so T defines a linear decision at each internal node, such that for each leaf v of T , either only red or only blue points lead the algorithm to v . The authors considered two measures of such a classifier, the

number of internal nodes and the depth of the tree, and prove a very strong negative result on high-dimensional classification trees: *Unless $NP=ZPP$, no polynomial-time algorithm for optimizing the depth of a classifier can have approximation ratio better than any fixed constant.* Das and Goodrich [4] showed that the following problem is NP-complete: *Given a set S of n points in \mathbb{R}^3 , divided in two concept classes red and blue, decide if there exists a decision tree T with at most k nodes that separates the red points from the blue points.*

5 Separability With Axis-Parallel Partitions

In this section, we consider k -level trees defined by axis-parallel lines. First we show how to compute a 2-level tree as in Figure 17(a). Suppose that ℓ_0 is vertical and ℓ_1, ℓ_2 are horizontal. Other cases, which also depend on the color assigned to the rectangles produced by the 2-level tree structure, can be handle analogously. A key observation is that if there exists a separating 2-level tree, then sweeping with a vertical line ℓ from left to right the sets of red and blue points on the left of ℓ have to be separable by an horizontal line until ℓ becomes ℓ_0 . It is used in the following $O(n)$ time algorithm.

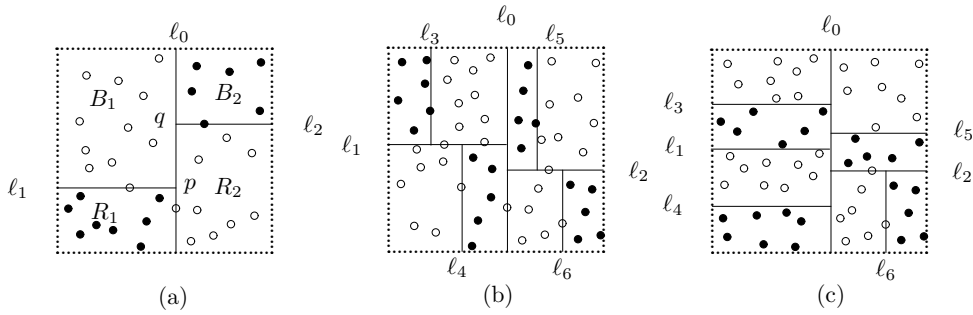


Figure 17: (a) Axis-parallel 2-level tree, (b) and (c) axis-parallel 3-level trees.

Axis-parallel 2-level tree algorithm. Compute the median M of the x -coordinates of the points. Let ℓ be the vertical line through M . Let R_1 and B_1 (resp. R_2 and B_2) be the subsets of red and blue points on the left (resp. right) side of ℓ . In $O(n/2)$ time check whether R_1 and B_1 (resp. R_2 and B_2) are line separable with a horizontal line. If the two answers are negative, stop the algorithm, there is not separating 2-level tree. If the two answers are affirmative, compute a separating 2-level tree in constant time with $\ell_0 = \ell$. Otherwise, assume that the affirmative answer is on the left of ℓ . In $O(n/2)$ compute the witnesses of the separability for R_1 and B_1 according to a sweep line with a horizontal line. Proceed recursively on the right of ℓ , computing the new median M' of the x -coordinates of the points in R_2 and B_2 , and updating the new witnesses obtained from the stored witnesses compared with the computed witnesses for the new subsets. The time needed is $O(n/2) + O(n/4) + O(n/8) + \dots = O(n)$. Notice that according to the vertical/horizontal choices for ℓ_0, ℓ_1 , and ℓ_2 , the number of different types of 2-level trees is 2^{2^2-1} . So the overall algorithm takes $O(n)$ time.

Theorem 5. *A separating axis-parallel 2-level tree for R and B can be computed in $\Theta(n)$ time.*

A *crossing 2-level tree* is a 2-level tree for R and B defined by two axis-parallel perpendicular lines ($p = q$). A separating crossing 2-level tree is also a *separating horizontal/vertical double-wedge* for R and B . In [1], the authors showed an $\Omega(n \log n)$ time lower bound for the horizontal/vertical double-wedge separability criterion, so this time lower bound applies to the crossing 2-level tree. An $O(n \log n)$ time algorithm for the separability by a crossing 2-level tree can be obtained by first sorting the points of $R \cup B$ by both x and y -coordinate and then applying an easy modification of the linear time algorithm above.

Axis-parallel 3-level tree algorithm. A key observation is that if there exists a 3-level tree for R and B with a configuration as in Figure 17(b), then for any vertical line ℓ_0 crossing the axis-parallel bounding box of $R \cup B$, either (i) the subset of points of $R \cup B$ on the left of ℓ_0 is separable by a 2-level tree or (ii) the subset of points of $R \cup B$ on the right of ℓ_0 is separable by a 2-level tree.

It is easy to see that we can do a binary search, moving left or right ℓ_0 in parallel, until both subsets of points of $R \cup B$ on the left and on the right of ℓ_0 become separable by a 2-level tree, getting an $O(n \log n)$ time algorithm. Other configurations can be handle analogously. Nevertheless, next we show a linear time algorithm for the axis-parallel 3-level tree.

Given a particular case, as in Figure 17(b), we can proceed as follows: In $O(n)$ we compute the vertical line ℓ'_3 containing the ray ℓ_3 using the median technique above such that the points on the left of ℓ'_3 are separable by a horizontal line, say ℓ'_1 ; we also compute a vertical interval I_1 where this horizontal line ℓ'_1 can be located. Then we proceed analogously (using the median technique) with the points on the right of the computed ℓ'_3 till find the vertical line ℓ'_4 containing the ray ℓ_4 such that the points between ℓ'_3 and ℓ'_4 are monochromatic, spending $O(n)$ time in this second process. Then we proceed analogously with the points on the right of ℓ'_4 till find the vertical line ℓ_0 such that the points between ℓ'_4 and ℓ_0 are separable by a horizontal line located in the computed vertical interval I_1 ; again we spend $O(n)$ time in this third process. Thus, the computation of the line ℓ_0 and the 2-level tree on the left of ℓ_0 takes $O(n)$ time. Similarly, in additional $O(n)$ time we check that the points on the right of the computed ℓ_0 are separable by a 2-level tree.

Notice that depending on the vertical/horizontal choices for $\ell_0, \ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6$, and the colors assigned to each region, the number of different types of axis-parallel 3-level trees is 2^{2^3-1} . So the overall algorithm takes $O(n)$ time.

Theorem 6. *A separating axis-parallel 3-level tree for R and B can be computed in $\Theta(n)$ time.*

Minimum axis-parallel level tree algorithm. The general problem consists of computing the minimum depth axis-parallel level tree for R and B . We can consider that the horizontal or vertical lines of any k -level tree for $R \cup B$ pass through a red or a blue point because we can move in parallel these lines (to the left the vertical lines, and to bottom the horizontal lines) till they bump a red or blue point. Thus, there are $2n$ possible horizontal and vertical lines for any level tree including the vertical and horizontal lines defining the bounding box of $R \cup B$.

Because of the general position assumption, at most two points are on the same vertical or horizontal line. Since we are computing binary space partitions of $R \cup B$, a point belongs

to only one part. Thus, a vertical (horizontal) line ℓ of a tree passes through either (1) one point p of $R \cup B$, or (2) exactly two points p, q of $R \cup B$. In case (1) we consider that the the point p belongs to the left (or below) part of the bipartition defined by ℓ ; in case (2) three possibilities have to be considered: p belongs to the left (or below) part and q belongs to the right (or above) part of the bipartition defined by ℓ , or vice versa, or both p and q belong to the left (or below) part. We sort the points of $R \cup B$ by both the x - and y -coordinate, assuming for simplicity and without loss of generality, that all the points from $R \cup B$ have different x - and y -coordinates.

We use dynamic programming to compute a minimum-level tree for $R \cup B$, and do another optimization as well, since a subproblem is simply an axis-parallel rectangle. There are $\binom{2n}{4} = O(n^4)$ possible rectangles or subproblems, each defined by four points. Next we show how to build a dynamic programming table that solves the problem.

In any solution to a problem P , if a vertical (horizontal) line ℓ is part of the solution, then it produces two subproblems, P_1 and P_2 , defined by the subsets of red and blue points on the left (below) and on the right (above) of ℓ . Therefore, we can consider the rectangles on each side of ℓ separately, compute the minimum-level tree for each of the two subproblems, and obtain the minimum-level tree for the problem, just adding the level (plus one) of the line ℓ . Thus, if $M(\cdot)$ denotes the number of levels of the minimum-level tree for P , and ℓ splits P into P_1 and P_2 , $P_1 \cap P_2 = \emptyset$, then

$$M(P) = \max\{M(P_1), M(P_2)\} + 1.$$

The $2n$ vertical and horizontal lines passing through a red or blue point of $R \cup B$ are linear separator candidates, along with the four lines of the bounding box \mathcal{B} of $R \cup B$. We sort these lines by x - and y -coordinates labelling them with the same label of the point they pass through according with their vertical or horizontal order.

A subproblem is a rectangle $\mathcal{R}((i, j), (k, l))$ defined by two opposite corners, the bottom-left corner (i, j) and the top-right corner (k, l) . The bottom-left corner (i, j) is the intersection point of the vertical passing through point i and the horizontal line passing through point j . The top-right corner (k, l) is the intersection point of a vertical line passing through point k and the horizontal line passing through point l . All of these lines are from the above set of $2n$ lines. Let (x_p, y_p) denotes the coordinates of a point p . Then $\mathcal{R}((i, j), (k, l))$ is a rectangle if and only if $x_i \leq x_j \leq x_k$, $x_i \leq x_l \leq x_k$, $y_j \leq y_i \leq y_l$, and $y_j \leq y_k \leq y_l$.

If p is a point contained in $\mathcal{R}((i, j), (k, l))$, then the vertical line through p splits $\mathcal{R}((i, j), (k, l))$ into two rectangles: $\mathcal{R}_l((p_1, p_2), (p, p_3))$ and $\mathcal{R}_r((q_1, q_2), (q_3, q_4))$. The points p_1, p_2, p , and p_3 are the points in $\mathcal{R}((i, j), (k, l))$ on the left of the vertical line through p that define the bounding box of $\mathcal{R}_l((p_1, p_2), (p, p_3))$; and the points q_1, q_2, q_3 , and q_4 are the points in $\mathcal{R}((i, j), (k, l))$ on the right of the vertical line through p that define the bounding box of $\mathcal{R}_r((q_1, q_2), (q_3, q_4))$. In a similar way, a horizontal line through p splits $\mathcal{R}((i, j), (k, l))$ into two rectangles: $\mathcal{R}_b((p'_1, p'_2), (p, p'_3))$ and $\mathcal{R}_t((q'_1, q'_2), (q'_3, q'_4))$. The bounding box of each rectangle (subproblem) and the points that it contains can be computed in linear time.

Let $\mathcal{R}((i, j), (k, l))$ be a rectangle in the minimum-level tree for $R \cup B$. If the vertical line through p belongs to the minimum-level tree, then

$$M(\mathcal{R}((i, j), (k, l))) = \max\{M(\mathcal{R}_l((p_1, p_2), (p, p_3))), M(\mathcal{R}_r((q_1, q_2), (q_3, q_4)))\} + 1.$$

If the horizontal line through p belongs to the minimum-level tree, then

$$M(\mathcal{R}((i, j), (k, l))) = \max\{M(\mathcal{R}_b((p'_1, p'_2), (p, p'_3))), M(\mathcal{R}_t((q'_1, q'_2), (q'_3, q'_4)))\} + 1.$$

The dynamic programming table has $(n - 1)^2$ rows and $(n - 1)^2$ columns. The $(n - 1)^2$ rows correspond to possible bottom-left corners of rectangles defined by $n - 1$ vertical lines and $n - 1$ horizontal lines (we exclude the top-right corner of \mathcal{B}). The $(n - 1)^2$ columns correspond to possible top-right corners of rectangles defined by $n - 1$ vertical lines and $n - 1$ horizontal lines (we exclude the bottom-left corner of \mathcal{B}). Not all of the pairs of corners correspond to a rectangle; it depends whether the rectangle condition above holds. In the negative case we fill the rectangle with “false”; otherwise, we fill the rectangle with the value of the minimum-level tree for it. To do this, we need to know the values of the subproblems (rectangles) produced by any vertical or horizontal line passing through an interior point of the rectangle, thereby examining at most a linear number of different subproblems. We fill the dynamic programming table as follows: For each rectangle \mathcal{R} we fill its corresponding value $M(\mathcal{R})$ using the formula

$$M(\mathcal{R}) = \min_{j \in \mathcal{R}} \{\max\{M(\mathcal{R}_{l,j,v}), M(\mathcal{R}_{r,j,v})\}, \max\{M(\mathcal{R}_{b,j,h}), M(\mathcal{R}_{t,j,h})\}\} + 1,$$

where $\mathcal{R}_{l,j,v}$ and $\mathcal{R}_{r,j,v}$ ($\mathcal{R}_{b,j,h}$ and $\mathcal{R}_{t,j,h}$) are the subproblems obtained by splitting \mathcal{R} with the vertical (horizontal) line passing through the interior point $j \in \mathcal{R}$.

Thus, filling a rectangle of the table corresponds to first computing all of the red and blue points (say m points in total) in the rectangle sorted by both x - and y -coordinates in linear time, and then solving the at most $2m$ pairs of subproblems, which implies that the algorithm requires at most $4m$ lookups in the table for each of the $O(n^4)$ rectangles. Therefore, the total time and space required to solve the problem is $O(n^4 \cdot n) = O(n^5)$. In the same running time we can compute the set of vertical/horizontal lines of the minimum-level tree for R and B . Therefore, we get the following result.

Theorem 7. *A minimum separating axis-parallel level-tree for R and B can be computed in $O(n^5)$ time.*

Remark. A minimum-level separating tree using only vertical (or only horizontal) lines can easily be computed in $O(n \log n)$ time by considering color transitions in the x -sorted list of points $R \cup B$.

6 Conclusion

We have initiated a study of k -level linear classification trees. In future work, we may consider other k -level trees defined by cuts other than lines/hyperplanes, e.g., axis-aligned boxes or circles (Figure 18). Multilevel trees based on separation by circles or axis-aligned boxes have potential applications in bounding volume hierarchies for intersection detection.

Acknowledgments

D. Garijo and A. Márquez are supported by project MTM2008-05866-C03-01. E. Arkin and J. Mitchell are partially supported by the National Science Foundation (CCF-0729019, CCF-1018388). C. Seara is supported by projects MTM2009-07242 and Gen. Cat. DGR2009GR1040.

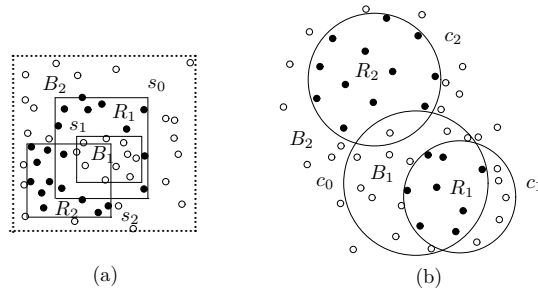


Figure 18: Example of 2-level trees based on (a) axis-aligned boxes, and (b) circles.

References

- [1] E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara, and S. S. Skiena. Some lower bounds on geometric separability problems. *International Journal of Computational Geometry and Applications* 16(1):1–26, 2006.
- [2] E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara, and S. S. Skiena. Some separability problems in the plane. *Abstracts of the 16th European Workshop on Computational Geometry*, 2000, pp. 51–54.
- [3] T. Asano, J. Hershberger, J. Pach, E. Sontag, D. Souvaine, and S. Suri. Separating bichromatic points by parallel lines. *Proc. 2nd Canadian Conference on Computational Geometry*, 1990, pp. 46–49.
- [4] G. Das and M. T. Goodrich. On the complexity of optimization problems for convex polyhedra and decision trees. *Computational Geometry: Theory and Applications* 8:123–137, 1997.
- [5] H. Edelsbrunner and F. P. Preparata. Minimum polygonal separation. *Information and Computation* 77:218–232, 1988.
- [6] S. Fekete. On the complexity of min-link red-blue separation problem. *Manuscript*, 1992.
- [7] M. Grigni, V. Mirelli, and C. H. Papadimitriou. On the difficulty of designing good classifiers. *SIAM Journal on Computing* 30(1):318–323, 2000.
- [8] F. Hurtado, M. Mora, P. A. Ramos, and C. Seara. Separability by two lines and by nearly-straight polygonal chains. *Discrete Applied Mathematics* 144:110–122, 2004.
- [9] F. Hurtado, M. Noy, P. A. Ramos, and C. Seara. Separating objects in the plane by wedges and strips. *Discrete Applied Mathematics* 109:109–138, 2000.
- [10] N. Megiddo. Linear-time algorithms for linear programming in \mathbb{R}^3 and related problems. *SIAM Journal on Computing* 12(4):759–776, 1983.
- [11] F. P. Preparata and M. I. Shamos. *Computational Geometry, An Introduction*. Springer-Verlag, 1988.