Active Learning a Convex Body in Low Dimensions*

Sariel Har-Peled† Mitchell Jones‡ Saladi Rahul§

May 20, 2022

Abstract

Consider a set $P \subseteq \mathbb{R}^d$ of n points, and a convex body C provided via a separation oracle. The task at hand is to decide for each point of P if it is in C using the fewest number of oracle queries. We show that one can solve this problem in two and three dimensions using $O(\varhexagon_P \log n)$ queries, where $\varhexagon_P$ is the largest subset of points of P in convex position. Furthermore, we show that in two dimensions one can solve this problem using $O(\circledcirc(P, C) \log^2 n)$ oracle queries, where $\circledcirc(P, C)$ is a lower bound on the minimum number of queries that any algorithm for this specific instance requires.

1. Introduction

1.1. Background

Active learning. Active learning is a subfield of machine learning, in which at any time, the learning algorithm is able to query an oracle for the label of a particular data point. One model for active learning is the membership query synthesis model [Ang87]. Here, the learner wants to minimize the number of oracle queries, as such queries are expensive—they usually correspond to either consulting with a specialist, or performing an expensive computation. In this setting, the learning algorithm is allowed to query the oracle for the label of any data point in the instance space. See [Set09] for a more in-depth survey on the various active learning models.

PAC learning. A classical approach for learning is using random sampling, where one gets labels for the samples (i.e., in the above setting, the oracle is asked for the labels of all items in the random sample). PAC learning studies the size of the sample needed. For example, consider the problem of learning a halfplane for n points $P \subset \mathbb{R}^2$, given a parameter $\varepsilon \in (0, 1)$. The first stage is to take a labeled random sample $R \subseteq P$. The algorithm computes any halfplane that classifies the sample correctly (i.e., the hypothesis). The misclassified points lie in the symmetric difference between the learned halfplane, and the (unknown) true halfplane, see Figure 1.1. In this case, the error region is a double wedge, and

*Research on this work was partially supported by NSF AF award CCF-1907400.

†Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@illinois.edu; http://sarielhp.org/.

‡Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; mfjones2@illinois.edu; http://mfjones2.web.engr.illinois.edu/.

§Department of Computer Science and Automation; Indian Institute of Science; Mathikere, Bengaluru, Karnataka 560012, India; saladi@iisc.ac.in; https://www.csa.iisc.ac.in/~saladi/.

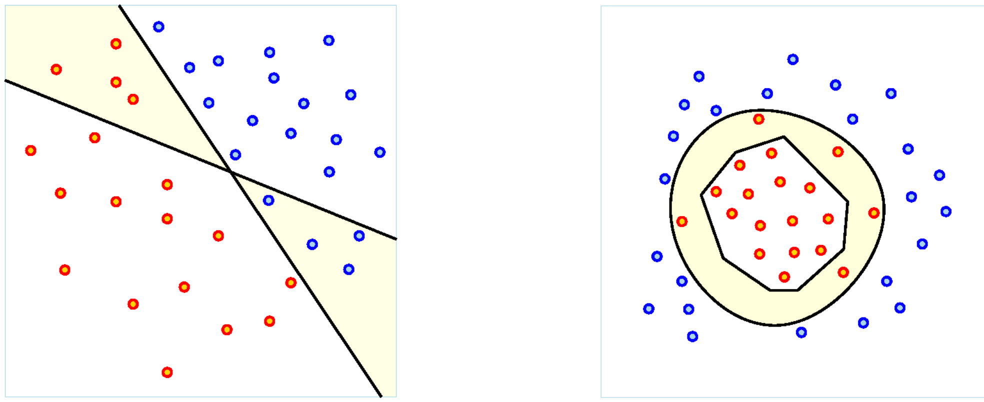
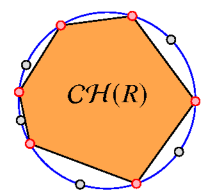


Figure 1.1: The shaded region shows the symmetric difference between the hypothesis and true classifier. (I) Learning halfspaces. (II) Learning arbitrary convex regions.

it is well known that its VC dimension [VC71] is a constant (at most eight). As such, by the ε-net Theorem [HW87], a sample of size $O(\varepsilon^{-1} \log \varepsilon^{-1})$ is an ε-net for double wedges, which implies that this random sampling algorithm has at most εn error.

A classical example of a hypothesis class that cannot be learned is the set of convex regions (even in the plane). Indeed, given a set of points P in the plane, any sample $R \subseteq P$ cannot distinguish between the true region being $\mathcal{CH}(R)$ or $\mathcal{CH}(P)$. Intuitively, this is because the hypothesis space in this case grows exponentially in the size of the sample (instead of polynomially).



Weak ε-nets. Because ε-nets for convex ranges do not exist, an interesting direction to overcome this problem is to define weak ε-nets [HW87]. A set of points R in the plane, not necessarily a subset of P, is a *weak ε-net* for P if for any convex body C containing at least εn points of P, it also contains a point of R. Matoušek and Wagner [MW03] gave a weak ε-net construction of size $O(\varepsilon^{-d}(\log \varepsilon^{-1})^{O(d^2 \log d)})$, which is doubly exponential in the dimension. The state of the art is the recent result of Rubin [Rub18], that shows a weak ε-net construction in the plane of size (roughly) $O(1/\varepsilon^{3/2})$. However, these weak ε-nets cannot be used for learning such concepts. Indeed, the analysis above required an ε-net for the symmetric difference of two convex bodies of finite complexity, see Figure 1.1.

1.2. Problem and motivation

The problem. In this paper, we consider a variation on the active learning problem, in the membership query synthesis model. Suppose that the learner is trying to learn an unknown convex body C in $\mathbb{R}^d$. Specifically, the learner is provided with a set P of n unlabelled points in $\mathbb{R}^d$, and the task is to label each point as either inside or outside C, see Figure 1.2. For a query $q \in \mathbb{R}^d$, the oracle either reports that $q \in C$, or returns a hyperplane separating q and C (as a proof that $q \notin C$). Note that if the query is outside the body, the oracle answer is significantly more informative than just the label of the point. The problem is to minimize the overall number of queries performed.

Hard and easy instances. Note that in the worst case, an algorithm may have to query the oracle for all input points—such a scenario happens when the input points are in convex position, and any possible subset of the points can be the points in the (appropriate) convex body. As such, the purpose

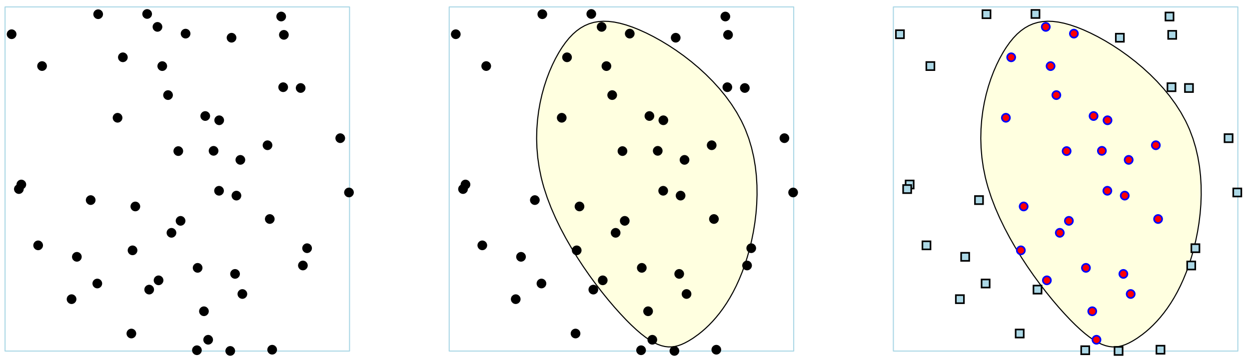
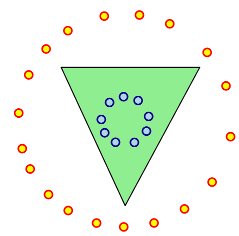


Figure 1.2: (I) A set of points P. (II) The unknown convex body C. (III) Classifying all points of P as either inside or outside C.

here is to develop algorithms that are *instance sensitive*—if the given instance is easy, they work well. If the given instance is hard, they might deteriorate to the naive algorithm that queries all points.

Natural inputs where one can hope to do better, are when relatively few points are in convex position. Such inputs are grid points, or random point sets, among others. However, there are natural instances of the problem that are easy, despite the input having many points in convex position. For example, consider when the convex body is a triangle, with the input point set being $n/2$ points spread uniformly on a tiny circle centered at the origin, while the remaining $n/2$ points are outside the convex body, spread uniformly on a circle of radius 10 centered at the origin. Clearly, such a point set can be classified using a constant number of oracle queries (in the best case). See Figure 3.1 for some related examples.



Additional motivation & some previous work.

(A) *Separation oracles.* The use of separation oracles is a common tool in optimization (e.g., solving exponentially large linear programs) and operations research. It is natural to ask what other problems can be solved efficiently when given access to this specific type of oracle.

(B) *Other types of oracles.* Various models of computation utilizing oracles have been previously studied within the community. Examples of other models include nearest-neighbor oracles (i.e., black-box access to nearest neighbor queries over a point set P) [HKMR16], and proximity probes (which given a convex polygon C and a query q, returns the distance from q to C) [PASG13]. It is reasonable to ask what classification-type problems can be solved with few oracle queries when using separation oracles.

Furthermore, other types of active learning models (in addition membership query model) have also been studied within the learning community, see, for example, [Ang87].

(C) *Active learning.* As discussed, the problem at hand can be interpreted as active learning a convex body in relation to a set of points P that need to be classified (as either inside or outside the body), where the queries are via a separation oracle. We are unaware of any work directly on this problem in the theory community, while there is some work in the machine learning community that studies related active learning classification problems [CAL94, GG07, Set09]. Specifically, Cohn et al. [CAL94] propose a similar problem to ours. For example, when the unknown body to be learned is an axis-parallel rectangle in the plane, they use neural networks to both learn and decide how to best query the oracle.

1.3. Our results

(A) We develop a greedy algorithm, for points in the plane, which solves the problem using $O(\hexagon_P \log n)$ oracle queries, where $\hexagon_P$ is the largest subset of points of P in convex position. See Theorem 2.8. It is known that for a random set of n points in the unit square, $\mathbf{E}[\hexagon_P] = \Theta(n^{1/3})$ [AB09], which readily implies that classifying these points can be solved using $O(n^{1/3} \log n)$ oracle queries. A similar bound holds for the $\sqrt{n} \times \sqrt{n}$ grid. An animation of this algorithm is on YouTube [HJR18]. We also show that this algorithm can be implemented efficiently, using dynamic segment trees, see Lemma 2.9.

(B) The above algorithm naturally extends to three dimensions, also using $O(\hexagon_P \log n)$ oracle queries. While the proof idea is similar to that of the algorithm in 2D, we believe the analysis in three dimensions is also technically interesting. See Theorem 2.18.

(C) For a given point set P and convex body C, we define the separation price $\circledcirc(P, C)$ of an instance (P, C), and show that any algorithm classifying the points of P in relation to C must make at least $\circledcirc(P, C)$ oracle queries (Lemma 3.1).

As an aside, we show that when P is a set of n points chosen uniformly at random from the unit square and C is a (fixed) smooth convex body, $\mathbf{E}[\circledcirc(P, C)] = O(n^{1/3})$, and this bound is tight when C is a disk (our result also generalizes to higher dimensions, see Lemma B.3). For randomly chosen points, the separation price is related to the expected size of the convex hull of $P \cap C$, which is also known to be $\Theta(n^{1/3})$ [Wei07]. We believe this result may be of independent interest, see Appendix B.

(D) In Section 3 we present an improved algorithm for the 2D case, and show that the number of queries made is $O(\circledcirc(P, C) \log^2 n)$. This result is $O(\log^2 n)$ approximation to the optimal solution, see Theorem 3.7.

(E) We consider the extreme scenarios of the problem: Verifying that all points are either inside or outside of C. For each problem we present a $O(\log n)$ approximation algorithm to the optimal strategy. The results are presented in Section 4, see Lemma 4.2 and Lemma 4.4.

(F) Section 5 presents an application of the above results, we consider the problem of minimizing a *convex* function $f : \mathbb{R}^3 \to \mathbb{R}$ over a point set P. Specifically, the goal is to compute $\arg\min_{p \in P} f(p)$. If f and its derivative can be efficiently evaluated at a given a query point, then f can be minimized over P using $O(\hexagon_P \log^2 n)$ queries to f (or its derivative) in expectation. We refer the reader to Lemma 5.5.

Given a set of n points P in $\mathbb{R}^d$, the discrete geometric median of P is a point $p \in P$ minimizing the function $\sum_{q \in P} \|p - q\|_2$. As a corollary of Lemma 5.5, we obtain an algorithm for computing the discrete geometric median for n points in the plane. The algorithm runs in $O(n \log^2 n \cdot (\log n \log\log n + \hexagon_P))$ expected time. See Lemma 5.6. In particular, if P is a set of n points chosen uniformly at random from the unit square, it is known $\mathbf{E}[\hexagon_P] = \Theta(n^{1/3})$ [AB09]—the discrete geometric median can be computed in $O(n^{4/3} \log^2 n)$ expected time.

While the discrete median is easy to approximate, we are unaware of any sub-quadratic algorithm for the discrete case even in the plane.

2. The greedy algorithm in two and three dimensions

2.1. Preliminaries

For a set of points $P \subseteq \mathbb{R}^2$, let $\mathcal{CH}(P)$ denote the convex hull of P. Given a convex body $C \subseteq \mathbb{R}^d$, two points $p, x \in \mathbb{R}^d \setminus \mathrm{int}(C)$ are ***mutually visible***, if the segment px does not intersect $\mathrm{int}(C)$, where $\mathrm{int}(C)$ is the interior of C. We also use the notation $P \cap C = \{p \in P \mid p \in C\}$.

For a point set $P \subseteq \mathbb{R}^d$, a ***centerpoint*** of P is a point $c \in \mathbb{R}^d$, such that for any closed halfspace h^+ containing c, we have $|h^+ \cap P| \geq |P|/(d+1)$. A centerpoint always exists, and it can be computed exactly in $O(n^{d-1} + n \log n)$ time [Cha04].

2.2. The greedy algorithm in 2D

2.2.1. Operations

Initially, the algorithm copies P into a set U of unclassified points. The algorithm is going to maintain an inner approximation $B \subseteq C$. There are two types of updates (Figure 2.1 illustrates the two operations):

(A) **expand**(p): Given a point $p \in C \setminus B$, the algorithm is going to:

  (i) Update the inner approximation: $B \leftarrow \mathcal{CH}(B \cup \{p\})$.

  (ii) Remove (and mark) newly covered points: $U \leftarrow U \setminus B$.

(B) **remove**(ℓ): Given a closed halfplane ℓ^+ such that $\mathrm{int}(C) \cap \ell^+ = \emptyset$, the algorithm marks all the points of $U_\ell = U \cap \mathrm{int}(\ell^+)$ as being outside C, and sets $U \leftarrow U \setminus U_\ell$.

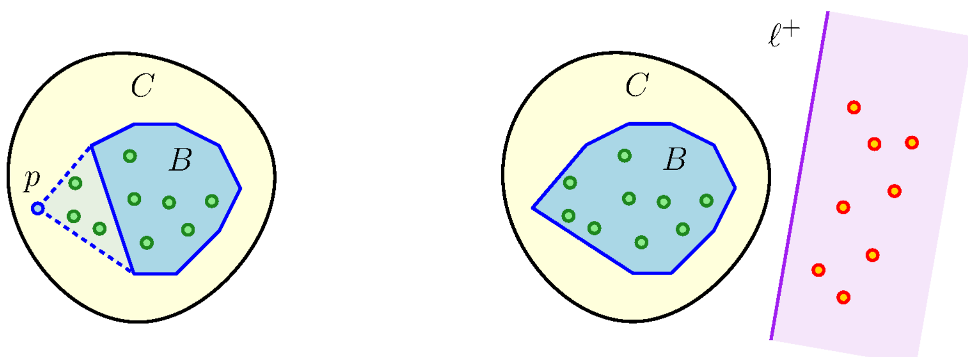


Figure 2.1: (I) Performing **expand**(p), and marking points inside C. (II) Performing **remove**(ℓ), and marking points outside C.

2.2.2. The algorithm

The algorithm repeatedly performs rounds, as described next, until the set of unclassified points is empty.

At every round, if the inner approximation B is empty, then the algorithm sets $U^+ = U$. Otherwise, the algorithm picks a line ℓ that is tangent to B with the largest number of points of U on the other side of ℓ than B. Let ℓ^- and ℓ^+ be the two closed halfspace bounded by ℓ, where $B \subseteq \ell^-$. The algorithm computes the point set $U^+ = U \cap \ell^+$. We have two cases:

A. If $|U^+| = O(1)$, then the algorithm queries the oracle for the status of each of these points. For every point $p \in U^+$, such that $p \in C$, the algorithm performs **expand**(p). Otherwise, the oracle returned a separating line ℓ, and the algorithm calls **remove**(ℓ^+).

B. Otherwise, the algorithm computes a centerpoint $c \in \mathbb{R}^2$ for U^+, and asks the oracle for the status of c. There are two possibilities:

  B.I. If $c \in C$, then the algorithm performs **expand**(c).

  B.II. If $c \notin C$, then the oracle returned a separating line h, and the algorithm performs **remove**(h).

2.2.3. Analysis

Let B_i be the inner approximation at the start of the ith iteration, and let z be the first index where B_z is not an empty set. Similarly, let U_i be the set of unclassified points at the start of the ith iteration, where initially $U_1 = U$.

Lemma 2.1. *The number of (initial) iterations in which the inner approximation is empty is* $z = O(\log n)$*.*

Proof: As soon as the oracle returns a point that is in C, the inner approximation is no longer empty. As such, we need to bound the initial number of iterations where the oracle returns that the query point is outside C. Let $f_i = |U_i|$, and note that $U_1 = P$ and $f_1 = |P| = n$. Let c_i be the centerpoint of U_i, which is the query point in the ith iteration (c_i is outside C). As such, the line separating c_i from C, returned by the oracle, has at least $f_i/3$ points of U_i on the same side as c_i, by the centerpoint property. All of these points get labeled in this iteration, and it follows that $f_{i+1} \leq (2/3)f_i$, which readily implies the claim, since $f_z < 1$, for $z = \lceil \log_{3/2} n \rceil + 1$. ■

Definition 2.2 (Visibility graph). Consider the graph G_i over U_i, where two points $p, r \in U_i$ are connected $\iff$ the segment pr does not intersect the interior of B_i.

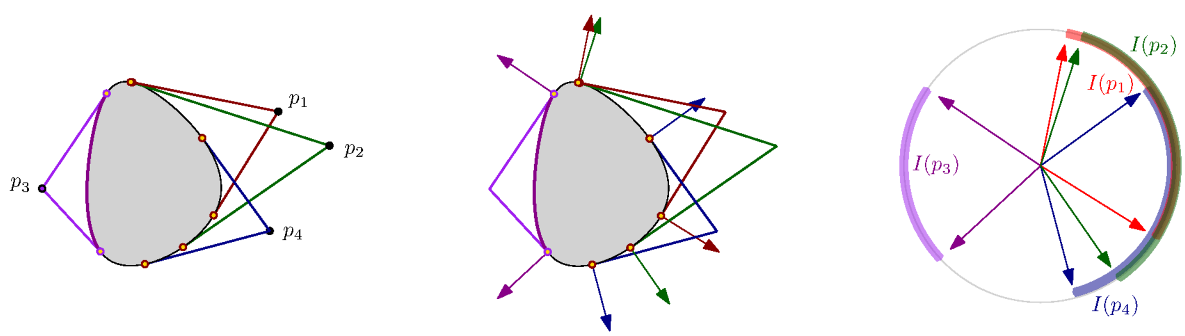


Figure 2.2

The visibility graph as an interval graph. For a point $p \in U_i$, let $I_i(p)$ be the set of all directions v (i.e., vectors of length 1) such that there is a line perpendicular to v that separates p from B_i. Formally, a line ℓ separates p from B_i, if the interior of B_i is on one side of ℓ and p is on the (closed) other side of ℓ (if $p \in \ell$, the line is still considered to separate the two). Clearly, $I_i(p)$ is a circular interval on the unit circle. See Figure 2.2. The resulting set of intervals is $\mathcal{V}_i = \{I_i(p) \mid p \in U_i\}$. It is easy to verify that the intersection graph of $\mathcal{V}_i$ is G_i. Throughout the execution of the algorithm, the inner approximation B_i grows monotonically, this in turn implies that the visibility intervals shrinks over time; that is, $I_i(p) \subseteq I_{i-1}(p)$, for all $p \in P$ and i. Intuitively, in each round, either many edges from G_i are removed (because intervals had shrunk and they no longer intersect), or many vertices are removed (i.e., the associated points are classified).

Definition 2.3. Given a set $\mathcal{V}$ of objects (e.g., intervals) in a domain D (e.g., unit circle), the ***depth*** of a point $p \in D$, is the number of objects in $\mathcal{V}$ that contain p. Let $\mathrm{depth}(\mathcal{V})$ be the maximum depth of any point in D.

When it is clear, we use $\mathrm{depth}(G)$ to denote $\mathrm{depth}(\mathcal{V})$, where $G = (\mathcal{V}, E)$ is the intersection graph in Definition 2.2.

First, we bound the number of edges in this visibility graph G and then argue that in each iteration, either many edges of G are discarded or vertices are removed (as they are classified).

Lemma 2.4. *Let $\mathcal{V}$ be a set of n intervals on the unit circle, and let $G = (\mathcal{V}, E)$ be the associated intersection graph. Then $|E| = O(\alpha\omega^2)$, where $\omega = \mathrm{depth}(\mathcal{V})$ and $\alpha = \alpha(G)$ is the size of the largest independent set in G. Furthermore, the upper bound on $|E|$ is tight.*

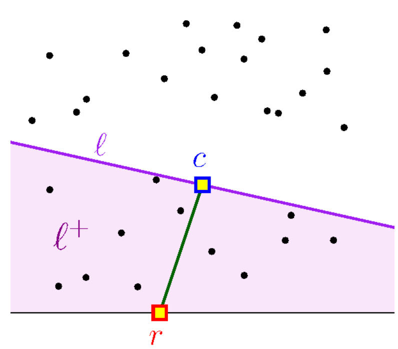
Proof: Let J be the largest independent set of intervals in G. The intervals of J divide the circle into $2|J|$ (atomic) circular arcs. Consider such an arc γ, and let $K(\gamma)$ be the set of all intervals of $\mathcal{V}$ that are fully contained in γ. All the intervals of $K(\gamma)$ are pairwise intersecting, as otherwise one could increase the size of the independent set. As such, all the intervals of $K(\gamma)$ must contain a common intersection point. It follows that $|K(\gamma)| \le \omega$.

Let $K'(\gamma)$ be the set of all intervals intersecting γ. This set might contain up to 2ω additional intervals (that are not contained in γ), as each such additional interval must contain at least one of the endpoints of γ. Namely, $|K'(\gamma)| \le 3\omega$. In particular, any two intervals intersecting inside γ both belong to $K'(\gamma)$. As such, the total number of edges contributed by $K'(\gamma)$ to G is at most $\binom{3\omega}{2} = O(\omega^2)$. Since there are $\le 2\alpha$ arcs under consideration, the total number of edges in G is bounded by $O(\alpha\omega^2)$, which implies the claim.

The lower bound is easy to see by taking an independent set of intervals of size α, and replicating every interval ω times. ■

Lemma 2.5. *Let P be a set of n points in the plane lying above the x-axis, c be a centerpoint of P, and $S = \binom{P}{2}$ be set of all segments induced by P. Next, consider any point r on the x-axis. Then, the segment cr intersects at least $n^2/36$ segments of S.*

Proof: If the segment cr intersects the segment p_1p_2, for $p_1, p_2 \in P$, then we consider p_1 and p_2 to no longer be mutually visible. It suffices to lower bound the number of pairs of points which lose mutual visibility of each other.



Consider a line ℓ passing through the point c. Let ℓ^+ be the closed halfspace bounded by ℓ containing r. Note that $|P \cap \ell^+| \geq n/3$, since c is a centerpoint of P, and $c \in \ell$. Rotate ℓ around c until there are $\geq n/6$ points on each side of rc in the halfspace ℓ^+. To see why this rotation of ℓ exists, observe that the two halfspaces bounded by the line spanning rc, have zero points on one side, and at least $n/3$ points on the other side — a continuous rotation of ℓ between these two extremes, implies the desired property.

Observe that points in ℓ^+ and on opposite sides of the segment cr cannot see each other, as the segment connecting them must intersect cr. Consequently, the number of induced segments that cr intersects is at least $n^2/36$. ∎

Lemma 2.6. *Let G_i be the intersection graph, in the beginning of the ith iteration, and let $m_i = |E(G_i)|$. After the ith iteration of the greedy algorithm, we have $m_{i+1} \leq m_i - \omega^2/36$, where $\omega = \mathrm{depth}(G_i)$.*

Proof: Recall that in the algorithm $U^+ = U_i \cap \ell^+$ is the current set of unclassified points and ℓ is the line tangent to B_i, where ℓ^+ is the closed halfspace that avoids the interior of B_i and contains the largest number of unlabeled points of U_i. We have that $\omega = |U^+|$.

If a **remove** operation was performed in the ith iteration, then the number of points of U^+ which are discarded is at least $\omega/3$. In this case, the oracle returned a separating line h between a centerpoint c of U^+ and the inner approximation. For the halfspace h^+ containing c, we have $t_i = |U^+ \cap h^+| \geq |U^+|/3 \geq \omega/3$. Furthermore, all the points of U^+ are pairwise mutually visible (in relation to the inner approximation B_i). Namely, $m_{i+1} = |E(G_i - (U^+ \cap h^+))| \leq m_i - \binom{t_i}{2} \leq m_i - \omega^2/36$.

If an **expand** operation was performed, the centerpoint c of U^+ is added to the current inner approximation B_i. Let r be a point in $\ell \cap B_i$, and let c_i be the center point of U_i computed by the algorithm. By Lemma 2.5 applied to r, c and U^+, we have that at least $\omega^2/36$ pairs of points of U^+ are no longer mutually visible to each other in relation to B_{i+1}. We conclude, that at least $\omega^2/36$ edges of G_i are no longer present in G_{i+1}. ∎

Definition 2.7. A subset of points $X \subseteq P \subseteq \mathbb{R}^2$ are in ***convex position***, if all the points of X are vertices of $\mathcal{CH}(X)$ (note that a point in the middle of an edge is not considered to be a vertex). The ***index*** of P, denoted by $\bigcirc_P$, is the cardinality of the largest subset of P of points which are in convex position.

Theorem 2.8. *Let C be a convex body provided via a separation oracle, and let P be a set of n points in the plane. The greedy classification algorithm performs $O\big((\bigcirc_P + 1)\log n\big)$ oracle queries. The algorithm correctly identifies all points in $P \cap C$ and $P \setminus C$.*

Proof: By Lemma 2.1, the number of iterations (and also queries) in which the inner approximation is empty is $O(\log n)$, and let $z = O(\log n)$ be the first iteration such that the inner approximation is not empty. It suffices to bound the number of queries made by the algorithm after the inner approximation becomes non-empty.

For $i \geq z$, let $G_i = (U_i, E_i)$ denote the visibility graph of the remaining unclassified points U_i in the beginning of the ith iteration. Any independent set in G_i corresponds to a set of points $X \subseteq P$ that do not see each other due to the presence of the inner approximation B_i. That is, X is in convex position, and furthermore $|X| \leq \bigcirc_P$.

For $0 \leq t \leq n$, let $s(t)$ be the first iteration i, such that $\mathrm{depth}(G_i) \leq t$. Since the depth of G_i is a monotone decreasing function, this quantity is well defined. An ***epoch*** is a range of iterations between $s(t)$ and $s(t/2)$, for any parameter t. We claim that an epoch lasts $O(\bigcirc_P)$ iterations (and every iteration issues only one oracle query). Since there are only $O(\log n)$ (non-overlapping) epochs till the algorithm terminates, as the depth becomes zero, this implies the claim.

So consider such an epoch starting at $i = s(t)$. We have $m = m_i = |E(G_i)| = O(\text{⬡}_P t^2)$, by Lemma 2.4, since ⬡_P is an upper bound on the size of the largest independent set in G_i. By Lemma 2.6, as long as the depth of the intervals is at least $t/2$, the number of edges removed from the graph at each iteration, during this epoch, is at least $\Omega(t^2)$. As such, the algorithm performs at most $O(m_i/t^2) = O(\text{⬡}_P)$ iterations in this epoch, till the maximum depth drops to $t/2$. ■

2.2.4. Implementing the greedy algorithm

With the use of dynamic segment trees [MN90] we show that the greedy classification algorithm can be implemented efficiently.

Lemma 2.9. *Let C be a convex body provided via a separation oracle, and let P be a set of n points in the plane. If an oracle query costs time T, then the greedy algorithm can be implemented in $O\big(n \log^2 n \log\log n + T \cdot \text{⬡}_P \log n\big)$ expected time.*

Proof: The algorithm follows the proof of Theorem 2.8. We focus on efficiently implementing the algorithm once inner approximation is no longer empty. Let $U \subseteq P$ be the subset of unclassified points. By binary searching on the vertices of the inner approximation B, we can compute the collection of visibility intervals $\mathcal{V}$ for all points in U in $O(|U| \log m) = O(n \log n)$ time (recall that $\mathcal{V}$ is a collection of circular intervals on the unit circle). We store these intervals in a dynamic segment tree $\mathcal{T}$ with the modification that each node v in $\mathcal{T}$ stores the maximum depth over all intervals contained in the subtree rooted at v. Note that $\mathcal{T}$ can be made fully dynamic to support updates in $O(\log n \log\log n)$ time [MN90].

An iteration of the greedy algorithm proceeds as follows. Start by collecting all points $U^+ \subseteq U$ realizing the maximum depth using $\mathcal{T}$. When $t = |U^+|$, this step can be done in $O(\log n + t)$ time by traversing $\mathcal{T}$. We compute the centerpoint of U^+ in $O(t \log t)$ expected time [Cha04] and query the oracle using this centerpoint. Either points of U are classified (and we delete their associated intervals from $\mathcal{T}$) or we improve the inner approximation. The inner approximation (which is the convex hull of query points inside the convex body C) can be maintained in an online fashion with insert time $O(\log n)$ [PS85, Chapter 3]. When the inner approximation expands, the points of U^+ have their intervals shrink. As such, we recompute $I(p)$ for each $p \in U^+$ and reinsert $I(p)$ into $\mathcal{T}$.

As defined in the proof of Theorem 2.8, an epoch is the subset of iterations in which the maximum depth is in the range $[t/2, t]$, for some integer t. During such an epoch, we make two claims:

(i) there are $\sigma = O(n)$ updates to $\mathcal{T}$, and
(ii) the greedy algorithm performs $O(n/t)$ centerpoint calculations on sets of size $O(t)$.

Both of these claims imply that a single epoch of the greedy algorithm can be implemented in expected time $O(\sigma \log n \log\log n + n \log n + T \cdot \text{⬡}_P)$. As there are $O(\log n)$ epochs, the algorithm can be implemented in expected time $O(n \log^2 n \log\log n + T \cdot \text{⬡}_P \log n)$.

We now prove the first claim. Recall that we have a collection of intervals $\mathcal{V}$ lying on the circle of directions. Partition the circle into k atomic arcs, where each arc contains $t/10$ endpoints of intervals in $\mathcal{V}$. Note that $k = 20n/t = O(n/t)$. For each circular arc γ, let $\mathcal{V}_\gamma \subseteq \mathcal{V}$ be the set of intervals intersecting γ. As the maximum depth is bounded by t, we have that $|\mathcal{V}_\gamma| \leq t + t/10 = 1.1t$. In particular, if $G[\mathcal{V}_\gamma]$ is the induced subgraph of the intersection graph G, then $G[\mathcal{V}_\gamma]$ has at most $\binom{|\mathcal{V}_\gamma|}{2} = O(t^2)$ edges.

In each iteration, the greedy algorithm chooses a point in an arc γ (we say that γ is *hit*) and edges are only deleted from $G[\mathcal{V}_\gamma]$. The key observation is that an arc γ can only be hit $O(1)$ times before all points of γ have depth below $t/2$, implying that it will not be hit again until the next epoch. Indeed, each time γ is hit, the number of edges in the induced subgraph $G[\mathcal{V}_\gamma]$ drops by a constant factor (Lemma 2.6). Additionally, when $G[\mathcal{V}_\gamma]$ has less than $\binom{t/2}{2}$ edges then any point on γ has depth less than $t/2$. These two facts imply that an arc is hit $O(1)$ times.

When an arc is hit, we must reinsert $|\mathcal{V}_\gamma| = O(t)$ intervals into $\mathcal{T}$. In particular, over a single epoch, the total number of hits over all arcs is bounded by $O(k)$. As such, $\sigma = O(kt) = O(n)$.

For the second claim, each time an arc is hit, a single centerpoint calculation is performed. Since each arc has depth at most t and is hit a constant number of times, there are $O(k) = O(n/t)$ such centerpoint calculations in a single epoch, each costing expected time $O(t \log t)$. ∎

2.3. The greedy algorithm in 3D

Consider the 3D variant of the 2D problem: Given a set of points P in $\mathbb{R}^3$ and a convex body C specified via a separation oracle, the task at hand is to classify, for all the points of P, whether or not they are in C, using the fewest oracle queries possible.

The greedy algorithm naturally extends, where at each iteration i a plane e_i is chosen that is tangent to the current inner approximation B_i, such that it's closed halfspace (which avoids the interior of B_i) contains the largest number of unclassified points from the set U_i. If the queried centerpoint is outside, the oracle returns a separating plane and as such points can be discarded by the **remove** operation. Similarly, if the centerpoint is reported inside, then the algorithm calls the **expand** and updates the 3D inner approximation B_i.

2.3.1. Analysis

Following the analysis of the greedy algorithm in 2D, we (conceptually) maintain the following set of objects: For a point $p \in U_i$, let $d_i(p)$ be the set of all unit length directions $v \in \mathbb{R}^3$ such that a plane perpendicular to v separates p from B_i. Let $\mathcal{P}_i = \{ d_i(p) \mid p \in U_i \}$. A set of objects form a collection of ***pseudo-disks*** if the boundary of every pair of them intersect at most twice. The following claim shows that $\mathcal{P}_i$ is a collection of pseudo-disks on $\mathbb{S}$, where $\mathbb{S}$ is the sphere of radius one centered at the origin.

Lemma 2.10. *The set $\mathcal{P}_i = \{ d_i(p) \subseteq \mathbb{S} \mid p \in U_i \}$ is a collection of pseudo-disks.*

Proof: Fix two points $p, r \in U_i$ such that the boundaries of $d_i(p)$ and $d_i(r)$ intersect on $\mathbb{S}$. Let ℓ be the line in $\mathbb{R}^3$ passing through p and r. Consider any plane e such that ℓ lies on e. Since ℓ is fixed, e has one degree of freedom. Conceptually rotate e until becomes tangent to B_i at point u'. The direction of the normal to this tangent plane, is a point in $X = \partial d_i(p) \cap \partial d_i(r)$. Note that this works also in the other direction — any point in X corresponds to a tangent plane passing through ℓ. The family of planes passing through ℓ has only two tangent planes to C. It follows that $|X| = 2$. As such, any two regions in $\mathcal{P}_i$ intersect as pseudo-disks. ∎

We need the following two classical results that follows from the Clarkson-Shor [CS89] technique.

Lemma 2.11 (Proof in Appendix A.1)**.** *Let $\mathcal{P}$ be a collection of n pseudo-disks, and let $V_{\leq k}(\mathcal{A})$ be the set of all vertices of depth at most k in the arrangement $\mathcal{A} = \mathcal{A}(\mathcal{P})$. Then $|V_{\leq k}(\mathcal{A})| = O(nk)$.*

Lemma 2.12 (Proof Appendix A.2)**.** *Let $\mathcal{P}$ be a collection of n pseudo-disks. For two integers $0 < t \leq k$, a subset $X \subseteq \mathcal{P}$ is a* **(t, k)-tuple** *if (i) $|X| \leq t$, (ii) $\exists p \in \cap_{d \in X} d$, and (iii) $\mathrm{depth}(p, \mathcal{P}) \leq k$. Let $L(t, k, n)$ be the set of all $(\leq t, k)$-tuples of $\mathcal{P}$. Then $|L(t, k, n)| = O(ntk^{t-1})$.*

Lemma 2.13. *Let $G_i = (\mathcal{P}_i, E_i)$ be the intersection graph of the pseudo-disks of $\mathcal{P}_i$ (in the ith iteration). If $\mathcal{A}(\mathcal{P}_i)$ has maximum depth k, then $|E_i| = O(nk)$. Furthermore, $\alpha(G_i) = \Omega(n/k)$, where $\alpha(G_i)$ denotes the size of the largest independent set in G_i.*

Proof: The first claim readily follows from Lemma 2.12 — indeed, $|E_i| = L(2, k, n) = O(nk)$ — since every intersecting pair of pseudo-disks induces a corresponding $(2, k)$-tuple.

For the second part, Turán's Theorem states that any graph has an independent set of size at least $n/(d_{\text{avg}}(G_i) + 1)$, where $d_{\text{avg}}(G_i) = 2|E_i|/n \le ck$ is the average degree of G_i and c is some constant. It follows that $\alpha(G_i) \ge n/(ck + 1) = \Omega(n/k)$. ■

The challenge in analyzing the greedy algorithm in 3D is that mutual visibility between pairs of points is not necessarily lost as the inner approximation grows. As an alternative, consider the *hypergraph* $H_i = (\mathcal{P}_i, \mathcal{E}_i)$, where a triple of pseudo-disks $d_1, d_2, d_3 \in \mathcal{P}_i$ form a hyperedge $\{d_1, d_2, d_3\} \in \mathcal{E}_i \iff d_1 \cap d_2 \cap d_3 \neq \varnothing$ (this is equivalent to the condition that the corresponding triple of points span a triangle which does not intersect B_i).

As in the analysis of the algorithm in 2D, we first bound the number of edges in H_i and then argue that enough progress is made in each iteration.

Lemma 2.14. *Let $H_i = (\mathcal{P}_i, \mathcal{E}_i)$ be the hypergraph in iteration i, and let G_i be the corresponding intersection graph of $\mathcal{P}_i$. If $\mathcal{A}(\mathcal{P}_i)$ has maximum depth k, then $|\mathcal{E}_i| = O(\alpha(G_i)k^3)$.*

Proof: Lemma 2.13 implies that G_i has an independent set of size $\Omega(f_i/k)$, where $f_i = |\mathcal{P}_i|$. Lemma 2.12 implies that $|\mathcal{E}_i| \le |L(3, k, f_i)| = O(f_i k^2) = O(\alpha(G_i)k^3)$. ■

The following is a consequence of the Colorful Carathéodory Theorem [Bár82], see Theorem 9.1.1 in [Mat02].

Theorem 2.15. *Let P be a set of n points in $\mathbb{R}^d$ and c be the centerpoint of P. Let $S = \binom{P}{d+1}$ be the set of all $d+1$ simplices induced by P. Then for sufficiently large n, the number of simplices in S that contain c in their interior is at least $c_d n^{d+1}$, where c_d is a constant depending only on d.*

Next, we argue that in each iteration of the greedy algorithm, a constant fraction of the edges in H_i are removed. The following is the higher dimensional version of Lemma 2.5.

Lemma 2.16. *Let P be a set of n points in $\mathbb{R}^3$ lying above the xy-plane, c be the centerpoint of P and $T = \binom{P}{3}$ be the set of all triangles induced by P. Next, consider any point r on the xy-plane. Then the segment cr intersects at least $\Omega(n^3)$ triangles of T.*

Proof: Let $S = \binom{P}{d+1}$ be the set of all simplices induced by P. Theorem 2.15 implies that the centerpoint c is contained in n^4/c_1 simplices of S for some constant $c_1 > 1$. Let K be a simplex that contains c and observe the segment cr must intersect at least one of the triangular faces τ of K. As $K \in S$, charge this simplex K to the triangular face τ. Applying this counting to all the simplices containing c, implies that at least n^4/c_1 charges are made. On the other hand, a triangle τ can be charged at most $n - 3$ times (because a simplex can be formed from τ and one other additional point of P). It follows that cr intersects at least $(n^4/c_1)/(n-3) = \Omega(n^3)$ triangles of T. ■

Lemma 2.17. *In each iteration of the greedy algorithm, the number of edges in the hypergraph $H_i = (\mathcal{P}_i, \mathcal{E}_i)$ decreases by at least $\Omega(k^3)$, where k is the maximum depth of any point in $\mathcal{A}(\mathcal{P}_i)$.*

Proof: Recall that $U^+ = U_i \cap e^+$ is the current set of unclassified points and e is the plane tangent to B_i, where e^+ is the closed halfspace that avoids the interior of B_i and contains the largest number of unlabeled points. Note that $|U^+| \geq k$.

In a **remove** operation, arguing as in Lemma 2.6, implies that the number of points of U^+ which are discarded is at least $k/4$. Since all of the discarded points are in a halfspace avoiding B_i, it follows that all the triples they induce are in H_i. Namely, at least $\binom{k/4}{3} = \Omega(k^3)$ hyperedges get discarded.

In an **expand** operation, the centerpoint c of U^+ is added to the current inner approximation B_i. Since all of the points of U^+ lie above the plane e, applying Lemma 2.16 on U^+ with the centerpoint c and a point lying on the plane e inside the (updated) inner approximation, we deduce that at least $\Omega(k^3)$ hyperedges are removed. ∎

Theorem 2.18. *Let $C \subseteq \mathbb{R}^3$ be a convex body provided via a separation oracle, and let P be a set of n points in $\mathbb{R}^3$. The greedy classification algorithm performs $O\big((\bigcirc_P + 1)\log n\big)$ oracle queries. The algorithm correctly identifies all points in $P \cap C$ and $P \setminus C$.*

Proof: The proof is essentially the same as Theorem 2.8. Arguing as in Lemma 2.1 implies that there are at most $O(\log n)$ iterations (and thus also oracle queries) in which the inner approximation is empty.

Now consider the hypergraph $H_1 = (\mathcal{P}_1, \mathcal{E}_1)$ at the start of the algorithm execution. As the algorithm progresses, both vertices and hyperedges are removed from the hypergraph. Let $H_i = (\mathcal{P}_i, \mathcal{E}_i)$ denote the hypergraph in the ith iteration of the algorithm. Recall that $\mathcal{P}_i$ is a set of pseudo-disks associated with each of the points yet to be classified. Observe that any independent set of pseudo-disks in the corresponding intersection graph G_i corresponds to an independent set of points with respect to the inner approximation B_i, and as such is a subset of points in convex position. Therefore, the size of any such independent set is bounded by $\bigcirc_P$.

Let k_i denote the maximum depth of any vertex in the arrangement $\mathcal{A}(\mathcal{P}_i)$. Lemma 2.14 implies that $|\mathcal{E}_i| = O(\bigcirc_P k_i^3)$. Lemma 2.17 implies that the number of hyperedges in the ith iteration decreases by at least $\Omega(k_i^3)$. Namely, after $O(\bigcirc_P)$ iterations, the maximum depth is halved. It follows that after $O(\bigcirc_P \log n)$ iterations, the maximum depth is zero, which implies that all the points are classified. Since the algorithm performs one query per iteration, the claim follows. ∎

3. An instance-optimal approximation in two dimensions

Before discussing the improved algorithm, we present a lower bound on the number of oracle queries performed by any algorithm that classifies all the given points. We then present the the improved algorithm, which matches the lower bound up to a factor of $O(\log^2 n)$.

3.1. A lower bound

Given a set P of points in the plane, and a convex body C, the ***outer fence*** of P is a closed convex polygon F_{out} with minimum number of vertices, such that $C \subseteq F_{\text{out}}$ and $C \cap P = F_{\text{out}} \cap P$. Similarly, the ***inner fence*** is a closed convex polygon F_{in} with minimum number of vertices, such that $F_{\text{in}} \subseteq C$ and $C \cap P = F_{\text{in}} \cap P$. Intuitively, the outer fence separates $P \setminus C$ from ∂C, while the inner fence separates $P \cap C$ from ∂C. The ***separation price*** of P and C is

$$\circledcirc(P, C) = |F_{\text{in}}| + |F_{\text{out}}|,$$

where $|F|$ denotes the number of vertices of a polygon F. See Figure 3.1 for an example.

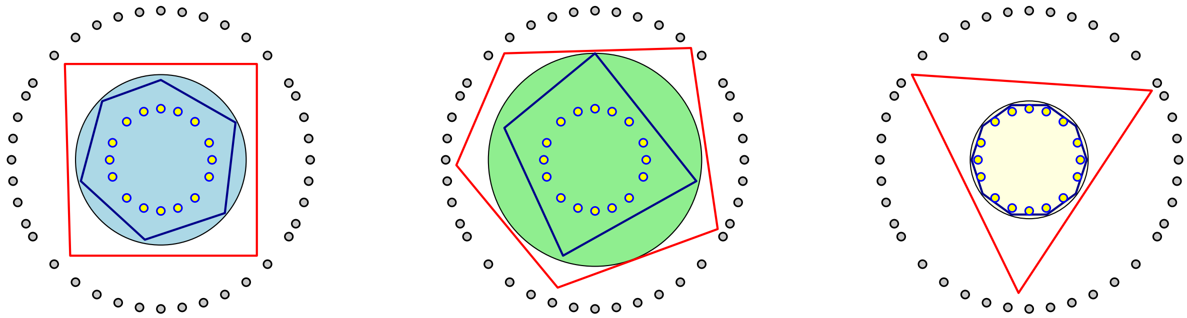


Figure 3.1: The separation price, for the same point set, is different depending on how “tight” the body is in relation to the inner and outer point set.

Lemma 3.1. *Given a point set P and a convex body C in the plane, any algorithm that classifies the points of P in relation to C, must perform at least $\circledcirc(P, C)$ separation oracle queries.*

Proof: Consider the set Q of queries performed by the optimal algorithm (for this input), and split it, into the points inside and outside C. The set of points inside, $Q_{\text{in}} = Q \cap C$ has the property that $Q_{\text{in}} \subseteq C$, and furthermore $\mathcal{CH}(Q_{\text{in}}) \cap P = C \cap P$ — otherwise, there would be a point of $C \cap P$ that is not classified. Namely, the vertices of $\mathcal{CH}(Q_{\text{in}})$ are vertices of a fence that separates the points of P inside C from the boundary of C. As such, we have that $|Q_{\text{in}}| \geq |\mathcal{CH}(Q_{\text{in}})| \geq |F_{\text{in}}|$.

Similarly, each query in $Q_{\text{out}} = Q \setminus Q_{\text{in}}$ gives rise to a separating halfplane. The intersection of the corresponding halfplanes is a convex polygon H which contains C, and furthermore contains no point of $P \setminus C$. Namely, the boundary of H behaves like an outer fence. As such, we have $|Q_{\text{out}}| \geq |H| \geq |F_{\text{out}}|$.

Combining, we have that $|Q| = |Q_{\text{in}}| + |Q_{\text{out}}| \geq |F_{\text{in}}| + |F_{\text{out}}| = \circledcirc(P, C)$, as claimed. ■

In Appendix B, we show that when P is a set of n points chosen uniformly at random from a square and C is a smooth convex body, $\mathbf{E}[\circledcirc(P, C)] = O(n^{1/3})$. Thus, when the points are randomly chosen, one can think of $\circledcirc(P, C)$ as growing sublinearly in n. Of course, for much more contrived instances, one would expect $\circledcirc(P, C)$ to be much smaller than $\hexagon_P$.

3.2. Useful operations

We start by presenting some basic operations that the new algorithm will use.

3.2.1. A directional climb

Given a direction v, a ***directional climb*** is a sequence of iterations, where in each iteration, the algorithm finds the extreme line perpendicular to v, that is tangent to the inner approximation B. The algorithm then performs an iteration with this line, as described in Section 2.2.2. See Figure 3.2 for an illustration. The directional climb ends when the outer halfspace induced by this line contains no unclassified point.

Claim 3.2. *A directional climb requires $O(\log n)$ oracle queries.*

Proof: Consider the tangent to B in the direction of v. At each iteration, we claim the number of points in this halfplane is reduced by a factor of $1/3$. Indeed, if the query (i.e., centerpoint) is outside C then at least a third of these points got classified as being outside. Alternatively, the tangent halfplanes moves in the direction of v, since the query point is inside C. But then the new halfspace contains at most $2/3$ fraction of the previous point set — again, by the centerpoint property. ■

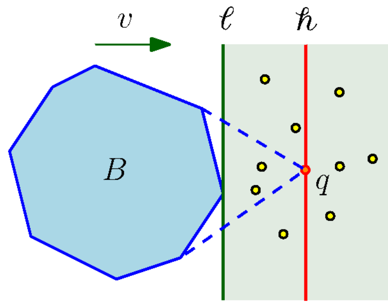


Figure 3.2: A directional climb. An iteration is done using the line ℓ. After updating B to include the query q, the algorithm chooses a new extreme line $\hbar$ tangent to B in the direction of v.

Figure 3.3: Unclassified points and their pockets.

3.2.2. Line cleaning

A ***pocket*** is a connected region of $\mathcal{CH}(U \cup B) \setminus B$, see Figure 3.3. For the set P of input points, consider the set of all lines

$$L(P) = \{\text{line}(p, r) \mid p, r \in P\} \tag{3.1}$$

they span. Let ℓ be a line that splits a pocket Υ into two regions, and furthermore, it intersects B. Let $I = \ell \cap \Upsilon$, and consider all the intersection points of interest along I in this pocket. That is,

$$\Xi(\Upsilon, \ell, P) = I \cap L(P) = \left\{ (\Upsilon \cap \ell) \cap \hbar \;\middle|\; \hbar \in L(P) \right\}.$$

In words, we take all the pairs of points of P (each such pair induces a line) and we compute the intersection points of these lines with the interval I of interest. Ordering the points of this set along ℓ, a prefix of them is in C, while the corresponding suffix are all outside C. One can easily compute this prefix/suffix by doing a binary search, using the separation oracle for C — see the lemma below for details. Each answer received from the oracle is used to update the point set, using **expand** or **remove** operations, as described in Section 2.2.1. We refer to this operation along ℓ as ***cleaning*** the line ℓ. See Figure 3.4.

Lemma 3.3. *Given a pocket Υ, and a splitting line ℓ, one can clean the line ℓ — that is, classify all the points of $\Xi = \Xi(\Upsilon, \ell, P)$ using $O(\log n)$ oracle queries. By the end of this process, Υ is replaced by two pockets, Υ_1 and Υ_2 that do not intersect ℓ. The pockets Υ_1 or Υ_2 may be empty sets.*

Proof: First, we describe the line cleaning procedure in more detail. The algorithm maintains, in the beginning of the ith iteration, an interval J_i on the line ℓ containing all the points of Ξ that are not classified yet. Initially, $J_1 = \Upsilon \cap \ell$. One endpoint, say $p_i \in J_i$ is on ∂B_i, and the other, say p'_i, is outside C, where B_i is the inner approximation in the beginning of the ith iteration.

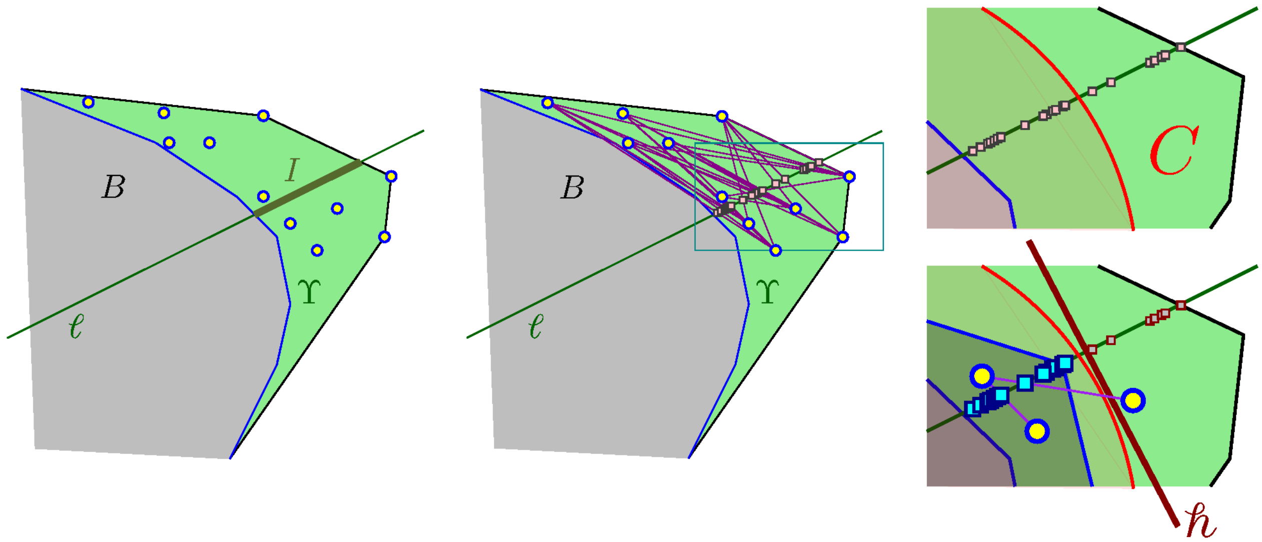


Figure 3.4: Line cleaning. All the intersection points of interest along ℓ are classified. The binary search results in the oracle returning a line $\hbar$ that separates the points outside from the points inside.

In the ith iteration, the algorithm computes the set $\Xi_i = J_i \cap \Xi$. If this set is empty, then the algorithm is done. Otherwise, it picks the median point u_i, in the order along ℓ in Ξ_i, and queries the oracle with u_i. There are two possibilities:

(A) If $u_i \in C$ then the algorithm sets $\Xi_{i+1} = \Xi_i \setminus [p_i, u_i)$, and $J_{i+1} = J_i \setminus [p_i, u_i)$.

(B) If $u_i \notin C$, then the oracle provided a closed halfspace h^+ that contains C. Let h^- be the complement open halfspace that contains u_i. The algorithm sets $\Xi_{i+1} = \Xi_i \setminus h^-$ and $J_{i+1} = J_i \cap h^+$.

This resolves the status of at least half the points in Ξ_i, and shrinks the active interval. The algorithm repeats this till Ξ_i becomes empty. Since $|\Xi| = O(n^2)$, this readily implies that the algorithm performs $O(\log n)$ iterations.

We now argue that the pocket is split — that is, Υ_1 and Υ_2 do not intersect ℓ. Assume that it is false, and let B' be the inner approximation after this procedure is done. Let L (resp. R) be the points of $U_\Upsilon = U \cap \Upsilon$ that are unclassified on one side (resp. other side) of ℓ. If the pocket is not split, then there are two points $p \in L$ and $r \in R$, such that $pr \cap B' = \emptyset$, and $\partial\mathcal{CH}(B' \cup L \cup R)$ intersects ℓ at the point $u = pr \cap \ell$. However, by construction, the point $u \in \Xi$. As such, the point u is now classified as either being inside or outside C, as it is a point in Ξ. If u is outside, then the halfplane h^- that classified it as such, must had classified either p or r as being outside C, which is a contradiction. The other option, is that u is classified as being inside, but then, it is in B', which is again a contradiction, as it implies that B' intersects the segment pr. ∎

3.2.3. Vertical pocket splitting

Consider a pocket Υ such that all of its points lie vertically above B, and the bottom of Υ is part of a segment of ∂B, see Figure 3.5. Such a pocket can be viewed as being defined by an interval on the x-axis corresponding to its two vertical walls. Let U_Υ be the set of unclassified points in this pocket. In each iteration, the algorithm computes the center point of U_Υ, and queries the separation oracle. As long as the query point is outside C, the algorithm performs a **remove** operation using the returned separating line.

When the oracle returns that the query point q is inside C, the algorithm computes the vertical line ℓ_q through q. The algorithm now performs line cleaning on this vertical line. This operation splits Υ

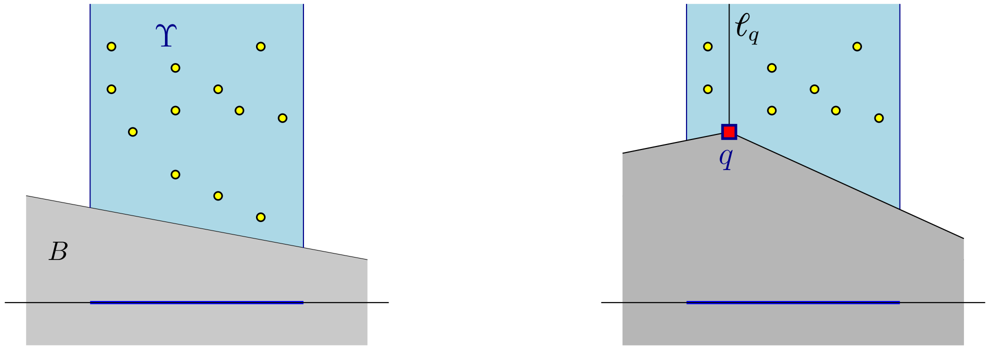


Figure 3.5: Vertical pocket splitting. The figure on the right is somewhat misleading — none of the unclassified points in the new pockets are mutually visible to each other after the line cleaning operation was done on the separating line.

into two sub-pockets. Crucially, since q was a centerpoint for U_Υ, the number of points in each of the two sub-pockets is at most $2|U_\Upsilon|/3$. See Figure 3.5.

3.3. The algorithm

The algorithm starts in the same way as the greedy algorithm of Section 2.2.2, until we obtain a non-empty inner approximation. The algorithm also maintains the convex hull of the unclassified points together with the inner approximation.

Next, the algorithm performs two directional climbs in the positive and negative directions of the x-axis. This uses $O(\log n)$ oracle queries and results in a computed segment $vv' \subseteq C$, where v, v' are vertices of the inner approximation B, such that all unclassified points lie in the vertical strip induced by these two points.

The algorithm now handles all points of U lying above vv' (the points below the line are handled in a similar fashion). Let B^+ be the set of vertices of B in the top chain. Note that B^+ consists of at most $O(\log n)$ vertices. For each vertex v of B^+, the algorithm performs line cleaning on the vertical line going through v. This results in $O(\log n)$ vertical pockets, where all vertical lines passing originally through B^+ are now clean.

The algorithm repeatedly picks a vertical pocket. If the pocket contains less three points the algorithm queries the oracle for the classification of these points, and continues to the next pocket. Otherwise, the algorithm performs a vertical pocket splitting operation, as described in Section 3.2.3. The algorithm stops when there are no longer any pockets (i.e., all the points above the segment vv' are classified). The algorithm then runs the symmetric procedure below this segment vv'.

3.4. Analysis

Lemma 3.4. *Given a point set P, and a convex polygon σ that is an inner fence for $P \cap C$; that is, $P \cap C \subseteq \sigma \subseteq C$. Then, there is a convex polygon π, such that*
(A) $P \cap C \subseteq \pi \subseteq \sigma$.
(B) $|\pi| \le 2|\sigma|$ (where $|Q|$ denotes the number of vertices of the polygon Q).
(C) Every edge of π lies on a line of $L(P)$, see Eq. (3.1).

Proof: Any edge e of σ that does not contain any point of P on it can be moved parallel to itself into the polygon until it passes through a point of P. Next, split the edges that contain only a single point of P, by adding this point as a vertex.

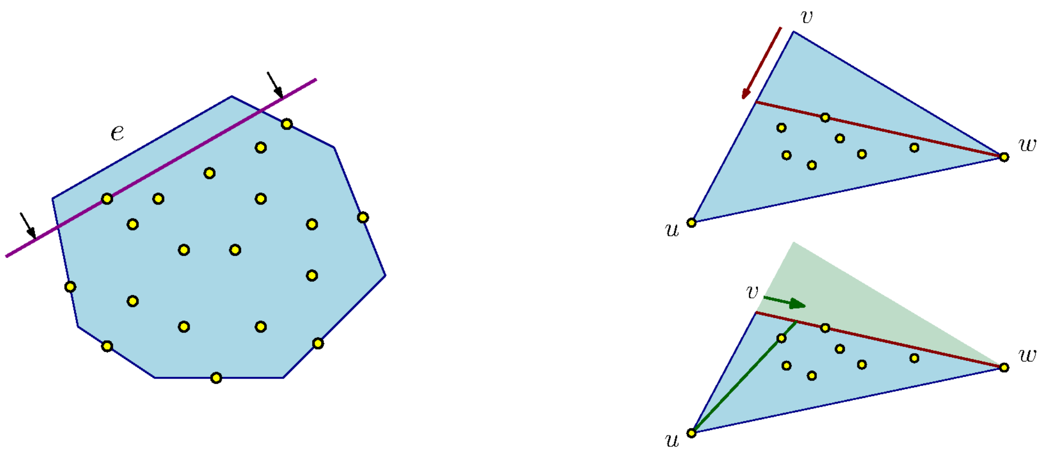


Figure 3.6: Constructing the polygon π from an inner fence σ.

Consider a vertex v of the polygon that is not in P — and consider the two adjacent vertices u, w, which must be in P. If $\triangle uvw \setminus uw$ contains no point of P, then we delete v from the polygon and replace it by the edge uw. Otherwise, move v towards u, until the edge vw hits a point of P. Next, move v towards w, till the edge vu hits a point of P. See Figure 3.6.

Repeating this process so that all edges contain two points of P means that properties (A) and (C) are met. Additionally, the number of edges of the new polygon π is at most twice the number of edges of σ, implying property (B). ■

Consider the inner and outer fences F_{in} and F_{out} of P in relation to C. Applying Lemma 3.4 to F_{in}, results in a convex polygon π that separates $P \cap C$ from ∂C, that has at most $2\,|F_{\text{in}}|$ vertices. Let V be the set of all vertices of the polygons $F_{\text{in}}, F_{\text{out}}$ and π.

The following two Lemmas state that if a vertical pocket Υ containing no vertex of V, then all points in Υ can be classified using $O(\log n)$ oracle queries. Finally, we analyze the scenario when Υ contains at least one vertex of V.

Lemma 3.5. *Let Υ be a vertical pocket created during the algorithm with current inner approximation B. Suppose that $V \cap \Upsilon = \varnothing$, then all points in $P \cap \Upsilon$ are outside C.*

Proof: Assume without loss of generality that Υ lies above B. Let $U = P \cap \Upsilon$ be the set of unclassified points in the pocket. Note that Υ is bounded by two vertical lines that were previously cleaned.

By assumption, Υ does not contain any vertex of π. It follows that there is a single edge of π which intersects the two vertical lines bounding Υ. Let u_L, u_R be these two intersection points, one lying on each line. By definition, we have $u_L, u_R \in C$. Furthermore, u_L, u_R lie on lines of $L(P)$ by construction of π. Since both vertical lines bounding Υ were cleaned, it must be that the segment $u_L u_R \subseteq B$. Since all points of U are above B, this implies that U lies above $u_L u_R$ and thus above π. Namely, all points of U are outside C. ■

Lemma 3.6. *Let Υ be a vertical pocket with $V \cap \Upsilon = \varnothing$. Then during the vertical pocket splitting operation of Section 3.2.3 applied to Υ, all oracle queries are outside C. In particular, all points of $P \cap \Upsilon$ are classified after $O(\log n)$ oracle queries.*

Proof: Let $U = P \cap \Upsilon$. By Lemma 3.5, all points of U lie outside C. Assume that the first statement of the Lemma is false, and let $U' \subseteq U$ be the set of unclassified points such that q was the centerpoint for

U' and $q \in C$. Now q is inside a triangle induced by three points of U'. Namely, there are (at least) two points outside C in this pocket that are not mutually visible to each other with respect to C. But this implies that F_{out} must have a vertex somewhere inside the vertical pocket Υ, which is a contradiction.

Hence, all oracle queries made by the algorithm are outside C. Each such query results in a constant reduction in the size of U, since the query point is a centerpoint of the unclassified points. It follows that after $O(\log |U|) = O(\log n)$ queries, all points in Υ are classified. ■

Theorem 3.7. *Let C be a convex body provided via a separation oracle, and let P be a set of n points in the plane. The improved classification algorithm performs $O\big(\big[1 + \circledcirc(P, C)\big] \log^2 n\big)$ oracle queries. The algorithm correctly identifies all points in $P \cap C$ and $P \setminus C$.*

Proof: The initial stage involves two directional climbs and $O(\log n)$ line cleaning operations, and thus requires $O(\log^2 n)$ queries.

A vertical pocket that contains a vertex of V is charged arbitrarily to any such vertex. Since the number of points in a pocket reduces by at least a factor of $1/3$ during a split operation, this means that a vertex of V is charged at most $O(\log n)$ times. Each time a vertex gets charged, it has to pay for the $O(\log n)$ oracle queries that were issued in the process of creating this pocket, and later on for the price of splitting it. Thus, we only have to account for queries performed in vertical pockets that do not contain a vertex of V. By Lemma 3.6, such a pocket will have all points inside it classified after $O(\log n)$ oracle queries.

However, the above implies that there are at most $O([1 + \circledcirc(P, C)] \log n)$ vertical pockets with no vertex of V throughout the algorithm execution. Since handling such a pocket requires $O(\log n)$ queries, the bound follows. ■

4. On emptiness variants in two dimensions

Here, we present two instance-optimal approximation algorithms for solving the following two variants:

(A) Emptiness: Find a point $p \in P \cap C$, or using as few queries as possible, verify that $P \cap C = \varnothing$.

(B) Reverse emptiness: Find a point $p \in P \setminus (P \cap C)$, or using as few queries as possible, verify that $P \cap C = P$.

For both variants we present $O(\log n)$ approximation (the algorithm for emptiness is randomized), improving over the general approximation algorithm of Section 3 which provides a $O(\log^2 n)$ approximation.

4.1. Emptiness: Are all the points outside?

Here we consider the problem of verifying that all the given points are outside the convex body.

Algorithm. The algorithm is a slight modification of the algorithm of Section 2.2.2. At each iteration the point set U^+ is the largest set of currently unclassified points in P contained in some halfspace tangent to the current inner approximation B. Let $\omega = |U^+|$. We make the following changes: If $\omega = O(1)$, test the membership of each point individually. Otherwise, choose a random point $q \in U^+$. If q is found to be inside C, we are done, as q is our witness. Otherwise q is outside, and a **remove** operation is performed. The algorithm then performs a regular iteration on U^+, as described in Section 2.2.2.

Analysis. Let G_i be the intersection graph (see Definition 2.2) over the points outside C in the beginning of the ith iteration. We need the following technical Lemma.

Lemma 4.1 (Proof in Appendix A.3)**.** *Suppose $P \cap C = \varnothing$. Then at any iteration i, the largest independent set in the visibility graph G_i is at most $|F_{\text{out}}|$.*

Lemma 4.2. *Let C be a convex body provided via a separation oracle, and let P be a set of n points in the plane. The randomized greedy classification algorithm for emptiness performs $O\big((|F_{\text{out}}| + 1)\log n\big)$ oracle queries with high probability. The algorithm always correctly verifies that $P \cap C = \varnothing$ or finds a witness point of P inside C.*

Proof: Suppose $P \cap C = \varnothing$. Then Lemma 4.1 along with the proof of Theorem 2.8 implies the result, by replacing the quantity $\bigcirc_P$ with $|F_{\text{out}}|$. If $P \cap C \neq \varnothing$, let U^+ be a set of points in the current iteration, $U^+_{\text{in}} = U^+ \cap C$, and $U^+_{\text{out}} = U^+_{\text{out}} \setminus U^+_{\text{in}}$. Observe that U^+_{in} remains the same throughout the algorithm execution, while U^+_{out} shrinks. If $\left|U^+_{\text{out}}\right| > |U^+|/2$, then by Lemma 2.6 the number of edges removed from G_i is $\Omega\Big(\left|U^+_{\text{out}}\right|^2\Big)$ (though the hidden constants will be smaller). Thus, after at most $O\big((|F_{\text{out}}|+1)\log n\big)$ iterations, we must encounter an iteration in which there is a set of points U^+ with $\left|U^+_{\text{out}}\right| < |U^+|/2$. Now the probability that our randomly sampled point lies in U^+_{in} is at least $1/2$. In particular, after an additional $O(\log n)$ iterations, the probability that we fail to find a witness point is at most $1/n^{\Omega(1)}$, thus implying the bound on the number of queries. ■

4.2. Reverse emptiness: Are all the points inside?

Here we consider the problem of verifying that all the given points are inside the convex body.

4.2.1. Algorithm

Initialization. Let $\mathcal{D} = \mathcal{CH}(P)$. Define $v, v' \in P$ to be the extreme left and right vertices of $\mathcal{D}$. Let v_1 and v_2 be the vertices adjacent to v on $\mathcal{D}$. Similarly define v'_1 and v'_2 for v'. The algorithm asks the oracle for the status of v, v_1, v_2, v', v'_1, and v'_2. If any of them are outside, the algorithm halts and reports the witness found. Otherwise, all points must lie either above or below the horizontal segment vv'. We now describe how to handle the points above vv' (the below case is handled similarly).

Let $\mathcal{D}^+$ be the polygonal chain which is $\mathcal{D}$ clipped inside region bounded by the segment vv' and two vertical lines passing through v and v'. Label the edges along $\mathcal{D}^+$ by $f_1, \ldots, f_k$ clockwise from v to v'. For $1 \leq i < j \leq k$, let ${\mathcal{D}^+}_{[i:j]}$ be the polygonal chain consisting of the consecutive edges $f_i, \ldots, f_j$. The algorithm now invokes the following recursive procedure.

Recursive procedure. A recursive call is described by two indices (i, j), the goal is to verify that all the points of P lying below ${\mathcal{D}^+}_{[i:j]}$ are inside C.

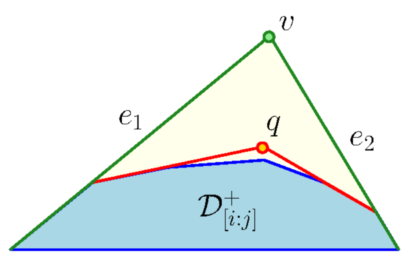
For a given recursive instance (i, j), the algorithm proceeds as follows. Begin by computing the lines ℓ_i and ℓ_j through the edges f_i and f_j respectively. Let $q = \ell_i \cap \ell_j$ be the point of intersection. The algorithm asks the oracle for the status of q. If q is inside, then all points below ${\mathcal{D}^+}_{[i:j]}$ must also be in C. The algorithm classifies the appropriate points and returns. Otherwise q is outside, and generates two recursive calls. Let $\ell = \lfloor (i + j)/2 \rfloor$ and $f_\ell = (x, y)$ be the middle edge in the chain ${\mathcal{D}^+}_{[i:j]}$. The algorithm queries the oracle with x and y. If either x or y is outside, the algorithm returns the appropriate witness found. Otherwise x and y are both inside. The algorithm recurses on the instances (i, ℓ) and (ℓ, j).

4.2.2. Analysis.

The analysis will use the polygon π, as defined in Lemma 3.4, applied to F_{in}. Specifically, it is an inner fence where $|\pi| = O(|F_{\text{in}}|)$ and every edge of π lies on a line of $L(P)$, see Eq. (3.1). Note that $\mathcal{D} \subseteq \pi$ and every edge of $\mathcal{D}$ lies on a line of $L(P)$. For each edge e of π, let $\ell_e \in L(P)$ be the line containing e. We can match every edge e of π with the edge $f(e)$ of $\mathcal{D}$ which lies on ℓ_e. If an edge f of $\mathcal{D}$ is matched to some edge of π, we say that f is ***active***. A recursive call (i, j) is ***alive*** if the query $q = \ell_i \cap \ell_j$ generated is outside C.

Lemma 4.3. *The number of alive recursive calls at the same recursive depth is at most* $|\pi| = O(|F_{\text{in}}|)$.

Proof: Fix an alive recursive call (i, j) with edges $f_i, \ldots, f_j$ of $\mathcal{D}$. Suppose that none of these edges are active. Because π is an inner fence for P and C, there must be a vertex v of π lying on or above the chain $\mathcal{D}^+{}_{[i:j]}$. Let e_1 and e_2 be the edges adjacent to v in π. For $\ell = 1, 2$, consider $f(e_\ell)$, the edge of $\mathcal{D}$ matched to e_ℓ. Since there are no active edges in $\mathcal{D}^+{}_{[i:j]}$, we have $f(e_\ell) \notin \{f_i, \ldots, f_j\}$ for $\ell = 1, 2$. This readily implies that all vertices in the polygonal chain $\mathcal{D}^+{}_{[i:j]}$ are contained in the wedge formed by v and the two edges e_1 and e_2.



In particular, the query q generated is inside π and thus C. Contradicting that the recursive call was alive. It follows that each alive recursive call must contain at least one active edge. The number of active edges is bounded by $|\pi|$, implying the result. ∎

Lemma 4.4. *Let C be a convex body provided via a separation oracle, and let P be a set of n points in the plane. The classification algorithm for reverse emptiness performs $O(|F_{\text{in}}| \log n)$ oracle queries. The algorithm correctly verifies that $P \cap C = P$ or finds a witness point of P outside C.*

Proof: Suppose all points of P are inside C. By Lemma 4.3, there are at most $O(|F_{\text{in}}|)$ alive recursive calls at each level of the recursion tree. Since the depth of the recursion tree is $O(\log n)$, the number of total alive recursive calls throughout the algorithm is $O(|F_{\text{in}}| \log n)$. At each alive recursive call of the above algorithm, $O(1)$ queries are made. This implies the result.

Otherwise not all points of P are inside C. At least one such point outside of C must be a vertex on the convex hull $\mathcal{D}$. Hence after at most $O(|F_{\text{in}}| \log n)$ oracle queries, this vertex will be queried and found to be outside C. ∎

5. Application: Minimizing a convex function

Suppose we are given a set of n points P in the plane and a convex function $f : \mathbb{R}^2 \to \mathbb{R}$. Our goal is to compute the point in P minimizing $\min_{p \in P} f(p)$. Given a point $p \in \mathbb{R}^2$, assuming that we can evaluate f and the derivative of f at p efficiently, we show that the point in P minimizing f can be computed using $O(\hexagon_P \log^2 n)$ evaluations to f or its derivative.

Definition 5.1. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. For a number $c \in \mathbb{R}$, define the ***level set of*** f as $\mathcal{L}_f(c) = \{p \in \mathbb{R}^d \mid f(p) \le c\}$. If f is a convex function, then $\mathcal{L}_f(c)$ is a convex set for all $c \in \mathbb{R}$.

Definition 5.2. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex (and possibly non-differentiable) function. For a point $p \in \mathbb{R}^d$, a vector $v \in \mathbb{R}^d$ is a ***subgradient*** of f at p if for all $q \in \mathbb{R}^d$, $f(q) \ge f(p) + \langle v, q - p\rangle$. The ***subdifferential*** of f at $p \in \mathbb{R}^d$, denoted by $\partial f(p)$, is the set of all subgradients $v \in \mathbb{R}^d$ of f at p.

It is well known that when the domain of f is $\mathbb{R}^d$ and f is a convex function, then $\partial f(p)$ is a non-empty set of all $p \in \mathbb{R}^d$ (for example, see [Fer13, Chapter 3]).

Let $\alpha = \min_{p\in P} f(p)$. We have that $\mathcal{L}_f(\alpha) \cap P = \{p \in P \mid f(p) = \alpha\}$ and $\mathcal{L}_f(\alpha') \cap P = \varnothing$ for all $\alpha' < \alpha$. Hence, the problem is reduced to determining the smallest value r such that $\mathcal{L}_f(r) \cap P$ is non-empty.

Lemma 5.3. *Let P be a collection of n points in the plane. For a given value r, let $C_r = \mathcal{L}_f(r)$. The set $C_r \cap P$ can be computed using $O(\bigcirc_P \log n)$ evaluations to f or its derivative. If T is the time needed to evaluate f or its derivative, the algorithm can be implemented in $O(n \log^2 n \log\log n + T \cdot \bigcirc_P \log n)$ expected time.*

Proof: The Lemma follows by applying Theorem 2.8. Indeed, let $C_r = \mathcal{L}_f(r)$ be the convex body of interest. It remains to design a separation oracle for C_r.

Given a query point $q \in \mathbb{R}^2$, first compute $c = f(q)$. If $c \le r$, then report that $q \in C_r$. Otherwise, $c > r$. In this case, compute some gradient vector v in $\partial f(q)$. Using the vector v, we can obtain a line ℓ tangent to the boundary of $\mathcal{L}_f(c)$ at q. As $\mathcal{L}_f(r) \subseteq \mathcal{L}_f(c)$, ℓ is a separating line for q and C_r, as desired.[1] As such, the number of separation oracle queries needed to determine $C_r \cap P$ is bounded by $O(\bigcirc_P \log n)$ by Theorem 2.8.

The implementation details of Theorem 2.8 are given in Lemma 2.9. ■

The algorithm. Let $\alpha = \min_{p\in P} f(p)$. For a given number $r \ge 0$, set $P_r = \mathcal{L}_f(r) \cap P$. We develop a randomized algorithm to compute α.

Set $P_0 = P$. In the ith iteration, the algorithm chooses a random point $p_i \in P_{i-1}$ and computes $r_i = f(p_i)$. Next, we determine P_{r_i} using Lemma 5.3. In doing so, we modify the separation oracle of Lemma 5.3 to store the collection of queries $S_i \subseteq P$ which satisfy $f(s) = r_i$ for all $s \in S_i$. We set $P_{i+1} = P_{r_i} \setminus S_i$. Observe that all points $p \in P_{i+1}$ have $f(p) < r_i$. The algorithm continues in this fashion until we reach an iteration j in which $|P_{j+1}| \le 1$. If $P_{j+1} = \{q\}$ for some $q \in P$, output q as the desired point minimizing the geometric median. Otherwise $P_{j+1} = \varnothing$, implying that $P_{r_j} = S_j$, and the algorithm outputs any point in the set S_j.

Analysis. We analyze the running time of the algorithm. To do so, we argue that the algorithm invokes the algorithm in Lemma 5.3 only a logarithmic number of times.

Lemma 5.4. *In expectation, the above algorithm terminates after $O(\log n)$ iterations.*

Proof: Let $V = \{f(p) \mid p \in P\}$ and $N = |V|$. For a number r, define $V_r = \{i \in V \mid i \le r\}$. Notice that we can reinterpret the algorithm described above as the following random process. Initially set $r_0 = \max_{i\in V} i$. In the ith iteration, choose a random number $r_i \in V_{r_{i-1}}$. This process continues until we reach an iteration j in which $|V_{r_j}| \le 1$.

[1]Note that q lies on ℓ. If we require that q lies in the interior of one of the halfspaces bounded by ℓ, we can shift ℓ infinitesimally to properly separate q and C_r.

We can assume without loss of generality that $V = \{1, 2, \ldots, N\}$. For an integer $i \le N$, let $T(i)$ be the expected number of iterations needed for the random process to terminate on the set $\{1, \ldots, i\}$. We have that $T(i) = 1 + \frac{1}{i-1} \sum_{j=1}^{i-1} T(i-j)$, with $T(1) = 0$. This recurrence solves to $T(i) = O(\log i)$. As such, the algorithm repeats this random process $O(\log N) = O(\log n)$ times in expectation. ■

Lemma 5.5. *Let P be a set of n points in $\mathbb{R}^2$ and let $f : \mathbb{R}^2 \to \mathbb{R}$ be a convex function. The point in P minimizing f can be computed using $O(\hexagon_P \log^2 n)$ evaluations to f or its derivative. The bound on the number of evaluations holds in expectation. If T is the time needed to evaluate f or its derivative, the algorithm can be implemented in $O(n \log^3 n \log\log n + T \cdot \hexagon_P \log^2 n)$ expected time.*

Proof: The result follows by combining Lemma 5.3 and Lemma 5.4. ■

5.1. The discrete geometric median

Let P be a set of n points in $\mathbb{R}^d$. For all $x \in \mathbb{R}^d$, define the function $f(x) = \sum_{q \in P-x} \|x - q\|_2$. The ***discrete geometric median*** is defined as the point in P minimizing the quantity $\min_{p \in P} f(p)$.

Note that f is convex, as it is the sum of convex functions. Furthermore, given a point p, we can compute $f(p)$ and the derivative of f at p in $O(n)$ time. As such, by Lemma 5.5, we obtain the following.

Lemma 5.6. *Let P be a set of points in $\mathbb{R}^2$. Then the discrete geometric median of P can be computed in $O(n \log^2 n \cdot (\log n \log\log n + \hexagon_P))$ expected time.*

Remark 5.7. For a set of n points P chosen uniformly at random from the unit square, it is known that in expectation $\hexagon_P = \Theta(n^{1/3})$ [AB09]. As such, the discrete geometric median for such a random set P can be computed in $O(n^{4/3} \log^2 n)$ expected time.

References

[AB09] G. Ambrus and I. Bárány. *Longest convex chains*. *Rand. Struct. & Alg.*, 35(2): 137–162, 2009.

[Ang87] D. Angluin. *Queries and concept learning*. *Machine Learning*, 2(4): 319–342, 1987.

[Bár82] I. Bárány. *A generalization of Carathéodory's theorem*. *Discrete Math.*, 40(2-3): 141–152, 1982.

[CAL94] D. A. Cohn, L. E. Atlas, and R. E. Ladner. *Improving generalization with active learning*. *Machine Learning*, 15(2): 201–221, 1994.

[Cha04] T. M. Chan. *An optimal randomized algorithm for maximum Tukey depth*. *Proc. 15th ACM-SIAM Sympos. Discrete Algs.* (SODA), 430–436, 2004.

[CS89] K. L. Clarkson and P. W. Shor. *Applications of random sampling in computational geometry, II*. *Discrete Comput. Geom.*, 4: 387–421, 1989.

[Dud74] R. M. Dudley. *Metric entropy of some classes of sets with differentiable boundaries*. *J. Approx. Theory*, 10(3): 227–236, 1974.

[Fer13] J. Ferrera. *An introduction to nonsmooth analysis*. Academic Press, 2013.

[GG07] Y. Guo and R. Greiner. *Optimistic active-learning using mutual information*. *Proc. 20th Int. Joint Conf. on AI* (IJCAI), 823–829, 2007.

[HJR18] S. Har-Peled, M. Jones, and S. Rahul. *An animation of the greedy classification algorithm in 2D*. `https://www.youtube.com/watch?v=IZX0VQdIgNA`. 2018.

[HKMR16] S. Har-Peled, N. Kumar, D. M. Mount, and B. Raichel. *Space exploration via proximity search*. *Discrete Comput. Geom.*, 56(2): 357–376, 2016.

[HW87] D. Haussler and E. Welzl. *ε-nets and simplex range queries*. *Discrete & Computational Geometry*, 2: 127–151, 1987.

[KLPS86] K. Kedem, R. Livne, J. Pach, and M. Sharir. *On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles*. *Discrete Comput. Geom.*, 1(1): 59–71, 1986.

[Mat02] J. Matoušek. *Lectures on discrete geometry*. Vol. 212. Grad. Text in Math. Springer, 2002.

[MN90] K. Mehlhorn and S. Näher. *Dynamic fractional cascading*. *Algorithmica*, 5(2): 215–241, 1990.

[MW03] J. Matoušek and U. Wagner. *New constructions of weak epsilon-nets*. *Proceedings of the nineteenth annual symposium on Computational geometry,* ACM. 129–135, 2003.

[PASG13] F. Panahi, A. Adler, A. F. van der Stappen, and K. Goldberg. *An efficient proximity probing algorithm for metrology*. *Int. Conf. on Automation Science and Engineering, CASE 2013,* 342–349, 2013.

[PS85] F. P. Preparata and M. I. Shamos. *Computational geometry - an introduction*. Texts and Monographs in Computer Science. Springer, 1985.

[Rub18] N. Rubin. *An improved bound for weak epsilon-nets in the plane*. *Proc. 59th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS)*,* 224–235, 2018.

[Set09] B. Settles. *Active Learning Literature Survey*. Tech. rep. #1648. Computer Science, Univ. Wisconsin, Madison, Jan. 2009.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. *On the uniform convergence of relative frequencies of events to their probabilities*. *Theory Probab. Appl.*, 16: 264–280, 1971.

[Wei07] *Random Polytopes, Convex Bodies, and Approximation*. *Stochastic Geometry: Lectures given at the C.I.M.E. Summer School held in Martina Franca, Italy, September 13–18, 2004*. Ed. by W. Weil. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 77–118.

A. Missing proofs

A.1. Proof of Lemma 2.11

The following is a standard consequence of the Clarkson-Shor [CS89] technique.

Restatement of Lemma 2.11. *Let $\mathcal{P}$ be a collection of n pseudo-disks, and let $V_{\leq k}(\mathcal{A})$ be the set of all vertices of depth at most k in the arrangement $\mathcal{A} = \mathcal{A}(\mathcal{P})$. Then $|V_{\leq k}(\mathcal{A})| = O(nk)$.*

Proof: Let $S \subseteq \mathcal{V}$ be a random sample where each pseudo-disk is independently placed into S with probability $1/k$. For each $p \in V_{\leq k}(\mathcal{A})$, let $\mathcal{E}_p$ be the event that p is a vertex in the union $\mathcal{U}(S)$ of this random subset of pseudo-disks. The probability that p is part of the union is at least the probability that both pseudo-disks defining p in $\mathcal{A}$ are sampled into S and the remaining $k-2$ objects containing p are not in S. Thus,

$$\mathbf{Pr}[\mathcal{E}_p] \geq \frac{1}{k^2}\left(1-\frac{1}{k}\right)^k \geq \frac{1}{e^2k^2},$$

since $1-1/x \geq e^{-2/x}$ for $x \geq 2$. If $|\mathcal{U}(S)|$ denotes the number of vertices on the boundary of the union, then linearity of expectations imply $\mathbf{E}[|\mathcal{U}(S)|] \geq |V_{\leq k}(\mathcal{A})|/(e^2k^2)$. On the other hand, it is well known the union complexity of a collection of n pseudo-disks is $O(n)$ [KLPS86]. Therefore, $\mathbf{E}[|\mathcal{U}(S)|] \leq \mathbf{E}[c\,|S|] \leq cn/k$, for some appropriate constant c. Putting both bounds on $\mathbf{E}[|\mathcal{U}(S)|]$ together, it follows that $cn/k \geq |V_{\leq k}(\mathcal{A})|/(e^2k^2) \iff |V_{\leq k}(\mathcal{A})| = O(nk)$. ■

A.2. Proof of Lemma 2.12

The following is a (slightly less) standard consequence of the Clarkson-Shor [CS89] technique.

Restatement of Lemma 2.12. *Let $\mathcal{P}$ be a collection of n pseudo-disks. For two integers $0 < t \leq k$, a subset $X \subseteq \mathcal{P}$ is a* **(t,k)-tuple** *if (i) $|X| \leq t$, (ii) $\exists p \in \cap_{d\in X} d$, and (iii)* $\mathrm{depth}(\mathrm{p},\mathcal{P}) \leq k$. *Let $L(t,k,n)$ be the set of all $(\leq t, k)$-tuples of $\mathcal{P}$. Then $|L(t,k,n)| = O(ntk^{t-1})$.*

Proof: Let $R \subseteq \mathcal{P}$ be a random sample, where each pseudo-disk is independently placed into R with probability $1/k$. Consider a specific (t,k)-tuple X, with a witness point p of depth $\leq k$. Without loss of generality, by moving p, one can assume p is a vertex of $\mathcal{A}(\mathcal{P})$.

Let $\mathcal{E}_X$ be the event that p is of depth exactly t in $\mathcal{A}(R)$, and $X \subseteq R$. For $\mathcal{E}_X$ to occur, all the objects of X need to be sampled into R, and each of the at most $k-t$ pseudo-disks containing p in its interior are not in R. Therefore

$$\mathbf{Pr}[\mathcal{E}_X] \geq \frac{(1-1/k)^{\mathrm{depth}(\mathrm{p},\mathcal{P})-|X|}}{k^{|X|}} \geq \frac{(1-1/k)^k}{k^t} \geq \frac{1}{e^2k^t}.$$

Note, that a vertex of depth $\leq k$ in $\mathcal{A}(R)$ corresponds to at most one such an event happening. We thus have, by linearity of expectations, that

$$\frac{|L(t,k,n)|}{e^2k^t} \leq \mathbf{E}\big[|V_{\leq t}(\mathcal{A}(R))|\big] = O(tn/k),$$

by Lemma 2.11. ■

A.3. Proof of Lemma 4.1

Restatement of Lemma 4.1. *Suppose $P \cap C = \varnothing$. Then at any iteration i, the largest independent set in the visibility graph G_i is at most $|F_{\mathrm{out}}|$.*

Proof: For the body C and point set P, define the set $R \subseteq P$ to be the maximum set of points such that no two points in R are visible with respect to C. Observe that R corresponds to the maximum independent set in the visibility graph for P with respect to the body C. We claim $|R| \leq |F_{\mathrm{out}}|$. Suppose that $|R| > |F_{\mathrm{out}}|$. Given the polygon F_{out}, for each edge e of F_{out} consider the line ℓ_e through e and let $\hbar_e^+$ be the halfspace bounded by ℓ_e which does not contain C in its interior. Then $\{\hbar_e^+ \mid e \in F_{\mathrm{out}}\}$ covers the space $\mathbb{R}^2 \setminus \mathrm{int}(C)$. By the hypothesis, one halfspace $\hbar_e^+$ must contain at least two points of R. But then these two such points are visible with respect to C, contradicting the definition of R.

We know that the size of the largest independent set (with respect to the current inner approximation B_i) is monotone increasing over the iterations. Hence each independent set can be of size at most $|R| \leq |F_{\mathrm{out}}|$. ■

B. Expected separation price for random points

We first extend the notion of separation price (see Section 3.1) to higher dimensions. For a closed convex d-dimensional polytope F, we let $f_k(F)$ denote the number of k-dimensional faces of F.

Definition B.1 (Separation price in higher dimensions). Let P be a set of points and C be a convex body in $\mathbb{R}^d$. The inner fence F_{in} is a closed convex d-dimensional polytope with the minimum number of vertices, such that $F_{\text{in}} \subseteq C$ and $C \cap P = F_{\text{in}} \cap P$. Similarly, the outer fence F_{out} is a closed convex d-dimensional polytope with the minimum number of facets, such that $C \subseteq F_{\text{out}}$ and $C \cap P = F_{\text{out}} \cap P$. The separation price is defined as $\circledcirc(P, C) = f_0(F_{\text{in}}) + f_{d-1}(F_{\text{out}})$.

By extending the argument of Lemma 3.1 to use Definition B.1, one can prove the following.

Lemma B.2. *Given a point set P and a convex body C in $\mathbb{R}^d$, any algorithm that classifies the points of P in relation to C, must perform at least $\circledcirc(P, C)$ separation oracle queries.*

Informally, for any fixed convex body C and a set of n points P chosen uniformly at random from the unit cube, the separation price is sublinear (approaching linear as the dimension increases).

Lemma B.3. *Let P be a set of n points chosen uniformly at random from the unit cube $[0, 1]^d$, and let C be a convex body in $\mathbb{R}^d$, with $\mathsf{vol}(C) \geq c$ for some constant $c \leq 1$. Then $\mathbf{E}[\circledcirc(P, C)] = O(n^{1-2/(d+1)})$, where O hides constants that depend on d and C.*

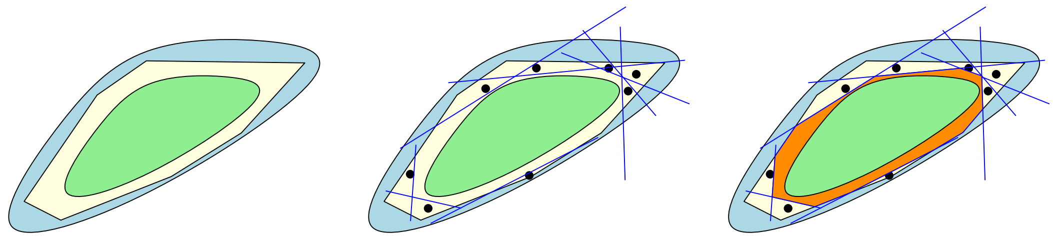
Proof: It is known that for convex bodies C, the expected number of vertices of the convex hull of $P \cap C$ is $O(n^{1-2/(d+1)})$. Indeed, since $\mathsf{vol}(C) \geq c$, the expected number of points of P which fall inside C is $m = \Theta(n)$ (and these bounds hold with high probability by applying any Chernoff-like bound). It is known that for m points chosen uniformly at random from C, the expected size of the convex hull of points inside C is $O(m^{1-2/(d+1)}) = O(n^{1-2/(d+1)})$ [Wei07]. This readily implies that $\mathbf{E}[f_0(F_{\text{in}})] = O(n^{1-2/(d+1)})$.

To bound $\mathbf{E}[f_{d-1}(F_{\text{out}})]$, we apply a result of Dudley [Dud74] which states the following. Given a convex body C and a parameter $\varepsilon > 0$, there exists a convex body D, which is a polytope formed by the intersection of $O(\varepsilon^{-(d-1)/2})$ halfspaces, such that $C \subseteq D \subseteq (1 + \varepsilon)C$, where $(1 + \varepsilon)C = \left\{p \in \mathbb{R}^d \;\middle|\; \exists q \in C : \|p - q\| \leq \varepsilon\right\}$.

We claim that the number of points of P which fall inside $D \setminus C$, plus the number of halfspaces defining D, is an upper bound on the size of the outer fence. Indeed, for each point p which falls in inside $D \setminus C$, let q be its nearest neighbor in C (naturally q lies on ∂C). Let $\hbar_p$ be the hyperplane which is perpendicular to the segment pq and passing through the midpoint of pq. Next, let $\hbar_p^+$ be the halfspace bounded by $\hbar_p$ such that $C \subseteq \hbar_p^+$. If H is the collection of $O(\varepsilon^{-(d-1)/2})$ halfspaces defining D, then it is easy to see that the polytope defined by

$$\left(\bigcap_{p \in P \cap (D \setminus C)} \hbar_p^+\right) \bigcap \left(\bigcap_{\hbar^+ \in H} \hbar^+\right)$$

separates the boundary of C from $P \setminus C$ (i.e., it is an outer fence). See the figure below.



We now bound the size of this inner fence. Since $\mathsf{vol}(D) - \mathsf{vol}(C) \le \mathsf{vol}((1+\varepsilon)C) - \mathsf{vol}(C) \le O(\varepsilon)$, we have that $\mathbf{E}[|P \cap (D \setminus C)|] = O(\varepsilon n)$. Combining both inequalities,

$$\mathbf{E}[f_{d-1}(F_{\text{out}})] \le \mathbf{E}[|P \cap (D \setminus C)|] + O(\varepsilon^{-(d-1)/2}) = O\left(\varepsilon n + \frac{1}{\varepsilon^{(d-1)/2}}\right).$$

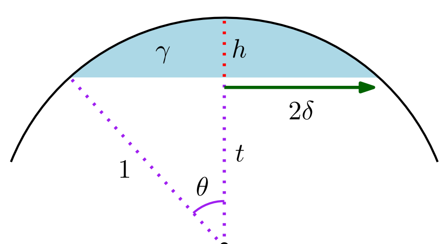
Choose $\varepsilon = 1/n^{2/(d+1)}$ to balance both terms, so that $\mathbf{E}[f_{d-1}(F_{\text{out}})] = O(n^{1-2/(d+1)})$. ∎

The next Lemma shows that the bound of Lemma B.3 is tight in the worst case.

Lemma B.4. *Let P be a set of n points chosen uniformly at random from the hypercube $[-2,2]^d$, and let C be a unit radius ball centered at the origin. Then $\mathbf{E}[\circledcirc(P,C)] \ge \mathbf{E}[f_0(F_{\text{in}})] = \Omega(n^{1-2/(d+1)})$, where Ω hides constants depending on d.*

Proof: For a parameter δ to be chosen, let $Q \subseteq \partial C$ be a maximal set of points such that:
(i) for any $p \in \partial C$, there is a point $q \in Q$ such that $\|p - q\| \le \delta$, and
(ii) for any two points $p, q \in Q$, $\|p - q\| \ge \delta$.
Note that $|Q| = \Omega(1/\delta^{d-1})$. For each $p \in Q$, we let γ_p be the spherical cap which is "centered" at p (in the sense that the center of the base of γ_p, p, and the origin are collinear) and has base radius 2δ. Let $\Gamma = \{\gamma_p \mid p \in Q\}$. By construction, the caps of Γ cover the surface of C.

By setting $\delta = 1/n^{1/(d+1)}$, we claim that for each cap $\gamma \in \Gamma$, in expectation $\Omega(1)$ points of P fall inside γ. This implies that there must be a vertex of the inner fence inside γ, and this holds for all caps in Γ. As such, the size of the inner fence is at least $|Q| = \Omega(1/\delta^{d-1}) = \Omega(n^{1-2/(d+1)})$.



To prove the claim, for all $\gamma \in \Gamma$, we show that $\mathsf{vol}(\gamma) = \Omega(1/n)$, and hence $\mathbf{E}[|P \cap \gamma|] = \Omega(1)$. By construction, the cap has a polar angle of $\theta = \Omega(\delta)$. Indeed, we have that $\theta \ge \sin(\theta) = 2\delta$ for $\theta \in [0, \pi/2]$ (which holds when n is sufficiently large). Let t denote the distance from the origin to the center of the base of γ. Then the height h of the spherical cap is $h = 1 - t = 1 - \cos(\theta) \ge \theta^2/6 = \Omega(\delta^2)$ (using the inequality $\cos(x) \le 1 - x^2/6$). Since the volume of the base of γ is $\Omega(\delta^{d-1})$, we have that $\mathsf{vol}(\gamma) = \Omega(h\delta^{d-1}) = \Omega(\delta^{d+1}) = \Omega(1/n)$, as required. ∎