Active Learning a Convex Body in Low Dimensions

Sariel Har-Peled
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
sariel@illinois.edu

Mitchell Jones
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
mfjones2@illinois.edu

Saladi Rahul
Dept. of Computer Science and Automation, Indian Institute of Science, Bangalore, India
saladi@iisc.ac.in

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 $O(\bigcirc_P \log n)$ queries, where $\bigcirc_P$ is the largest subset of points of P in convex position. In 2D, we provide an algorithm which efficiently generates these adaptive queries.

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. Finally, we consider other variations on the problem, such as using the fewest number of queries to decide if C contains all points of P.

As an application of the above, we show that the discrete geometric median of a point set P in $\mathbb{R}^2$ can be computed in $O(n \log^2 n \, (\log n \log\log n + \bigcirc_P))$ expected time.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Approximation algorithms, computational geometry, separation oracles, active learning

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.64

Category Track A: Algorithms, Complexity and Games

Related Version The full version of this paper is available at `https://arxiv.org/abs/1903.03693`.

Funding *Sariel Har-Peled*: Supported in part by NSF AF award CCF-1907400.
Mitchell Jones: Supported in part by NSF AF award CCF-1907400.

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 [2]. 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 [21] for a more in-depth survey on the various active learning models.

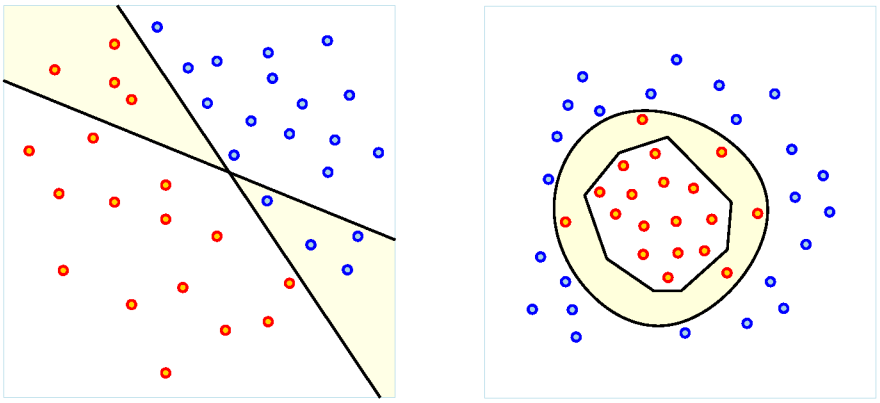


Figure 1.1 The shaded region shows the symmetric difference between the hypothesis and true classifier. (I) Learning halfspaces. (II) Learning arbitrary convex regions.

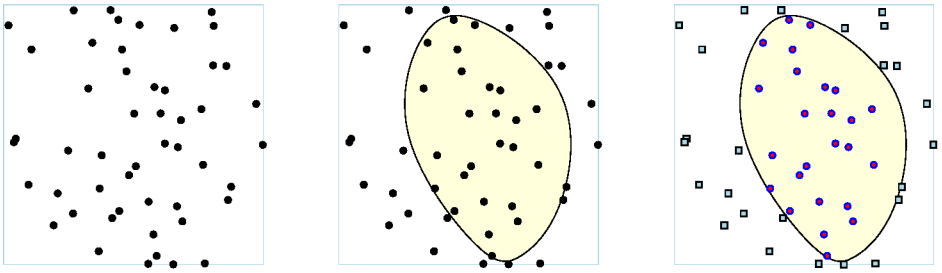
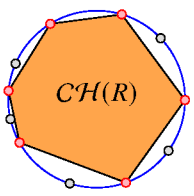


Figure 1.2 (I) A set of points P. (II) The unknown convex body C. (III) Classifying all points of P as either inside or outside C.

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 it is well known that its VC-dimension [22] is a constant (at most eight). As such, by the ε-net Theorem [13], 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).



We stress that the above argument does not necessarily imply these types of hypothesis classes are unlearnable in practice. In general, there are other ways for learning algorithms to handle hypothesis classes with high (or even infinite) VC-dimension (for example, using regularization or assuming there is a large margin around the decision boundary).

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 [13]. 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 [16] 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 [20], 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.

PAC learning with additional parameters

If one assumes the input instance obeys some additional structural properties, then random sampling can be used. For example, suppose that the point set P has at most k points in convex position. For an arbitrary convex body C, the convex hull $\mathcal{CH}(P \cap C)$ has complexity at most k. Let $R \subseteq P$ be a random sample, and C' be the learned classifier for R. The region of error is the symmetric difference between C and C'. In particular, since k-vertex polytopes in $\mathbb{R}^d$ have VC-dimension bounded by $O(d^2 k \log k)$ [15], this implies that the error region also has VC-dimension at most $O(d^2 k \log k)$. Hence if R is a random sample of size $O(d^2 k \log k \varepsilon^{-1} \log \varepsilon^{-1})$, the ε-net Theorem [13] implies that this sampling algorithm has error at most εn. However, even for a set of n points chosen uniformly at random from the unit square $[0,1]^2$, the expected number of points in convex position is $O(n^{1/3})$ [1]. Since we want $|R| < n$, this random sampling technique is only useful when ε is larger than $\log^2 n / n^{2/3}$ (ignoring constants).

To summarize the above discussions, random sampling on its own does not seem powerful enough to learn arbitrary convex bodies, even if one allows some error to be made. In this paper we focus on developing algorithms for learning convex bodies in low dimensions, where the algorithms are deterministic and do not make any errors.

1.2 Problem and motivation

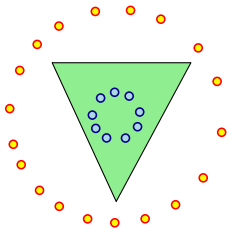
The problem

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 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 fully classified using a sequence of a constant number of oracle queries. See Figure 3.1 for some related examples.



1.3 Additional motivation & previous work

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. For example, Bárány and Füredi [3] study the problem of computing the volume of a convex body in $\mathbb{R}^d$ given access to a separation oracle.

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) [11], proximity probes (which given a convex polygon C and a query q, returns the distance from q to C) [18], and linear queries. Recently, Ezra and Sharir [7] gave an improved algorithm for the problem of point location in an arrangement of hyperplanes. Here, a *linear query* consists of a point x and a hyperplane h, and outputs either that x lies on h, or else which side of h contains x. Alternatively, their problem can be interpreted as querying whether or not a given point lies in a halfspace h^+. Here, we study the more general problem as the convex body can be the intersection of many halfspaces.

Furthermore, other types of active learning models (in addition membership query model) have also been studied within the learning community, see, for example, [2].

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 [6, 9, 21, 14].

For example, Kane et al. [14] study the problem of actively learning halfspaces with access to *comparison queries*. Given a halfspace h^+ to learn, the model has two types of queries: (i) label queries (given $x \in \mathbb{R}^d$, is $x \in h^+$?), and (ii) comparison queries (given $x_1, x_2 \in \mathbb{R}^d$, is x_1 closer to the boundary of h^+ than x_2?). For example, they show that in the plane, one can classify all points using $O(\log n)$ comparison/label queries in expectation.

1.4 Our results

Due to space constraints, not all of the results listed below are included in this version. We refer the reader to the full version of the paper [12] for proofs of missing results.

(A) We develop a greedy algorithm, for points in the plane, which solves the problem using $O(\bigcirc_P \log n)$ oracle queries, where $\bigcirc_P$ is the largest subset of points of P in convex position. See Theorem 8. It is known that for a random set of n points in the unit square, $\mathbf{E}[\bigcirc_P] = \Theta(n^{1/3})$ [1], 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 [10]. We also show that this algorithm can be implemented efficiently, using dynamic segment trees, see Lemma 9.
We remark that Kane et al. [14] develop a framework and randomized algorithm for learning a concept C, where the expected number of queries depends near-linearly on a parameter they define as the *inference dimension* [14, Definition III.1] of the concept class. For our problem, one can show that the inference dimension is $O(\bigcirc_P)$. As a corollary of their framework, one can obtain a randomized algorithm which solves our problem where the expected number of queries is $O(\bigcirc_P \log \bigcirc_P \log n)$. Our algorithm shaves a logarithmic factor in the number of queries and is deterministic.

(B) The above algorithm naturally extends to three dimensions, also using $O(\bigcirc_P \log n)$ 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 10.

(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 11).
As an aside, we show in [12] 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). 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 $\Theta(n^{1/3})$ [23]. We believe this result may be of independent interest/

(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 12.

(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 the full version of the paper [12].

(F) Section 4 presents an application of the above results, we consider the problem of minimizing a *convex* function $f : \mathbb{R}^2 \to \mathbb{R}$ 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 query point, then f can be minimized over P using $O(\bigcirc_P \log^2 n)$ queries to f (or its derivative) in expectation. We refer the reader to Lemma 17.
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 17, we obtain an algorithm for computing the discrete geometric median for n points in the plane. The algorithm runs in $O(n \log^2 n \cdot (\log n \log\log n + \bigcirc_P))$ expected time. See Lemma 18. In particular, if P is a set of n points chosen uniformly at random from the unit square, it is known $\mathbf{E}[\bigcirc_P] = \Theta(n^{1/3})$ [1] and hence the discrete geometric median can be computed in $O(n^{4/3} \log^2 n)$ expected time.
While there has been ample work on approximating the geometric median (recently, Cohen et al. [5] gave a $(1+\varepsilon)$-approximation algorithm to the geometric median in $O(dn \log^3(1/\varepsilon))$ time), we are unaware of any *exact* sub-quadratic algorithm for the discrete case even in the plane.

▶ Remark. Throughout this paper, the model of computation we have assumed is unit-cost real RAM.

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 $p, x \in \mathbb{R}^d \setminus \operatorname{int}(C)$ are **mutually visible**, if the segment px does not intersect $\operatorname{int}(C)$, where $\operatorname{int}(C)$ 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 [4].

Let C be a convex body in $\mathbb{R}^d$ and $q \in \mathbb{R}^d$ be a point such that q lies outside C. A hyperplane h **separates** q from C if q lies in the *closed* halfspace h^+ bounded by h, and C is contained in the *open* halfspace h^- bounded by h. This definition allows the separating hyperplane to contain the point q, and will simplify the descriptions of the algorithms.

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 $\operatorname{int}(C) \cap \ell^+ = \emptyset$, the algorithm marks all the points of $U_\ell = U \cap \operatorname{int}(\ell^+)$ as being outside C, and sets $U \leftarrow U \setminus U_\ell$.

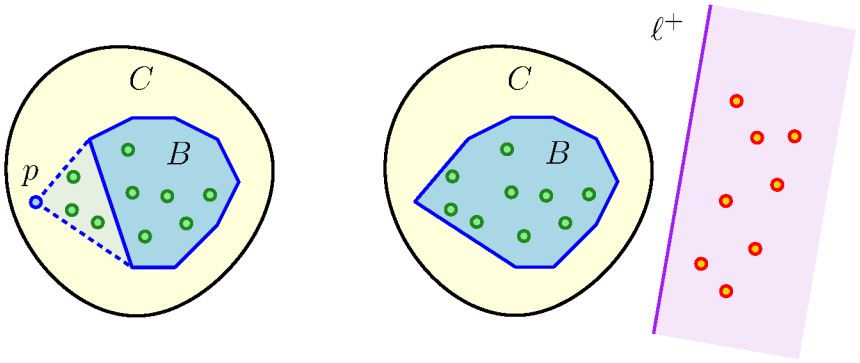


Figure 2.1 (I) Performing `expand`(p), and marking points inside C. (II) Performing `remove`(ℓ), and marking points outside C.

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) Suppose $|U^+|$ is of constant size. 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, $|U+|$ does not have constant size. 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:

- **(i)** If $c \in C$, then the algorithm performs `expand`(c).
- **(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 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** (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.*

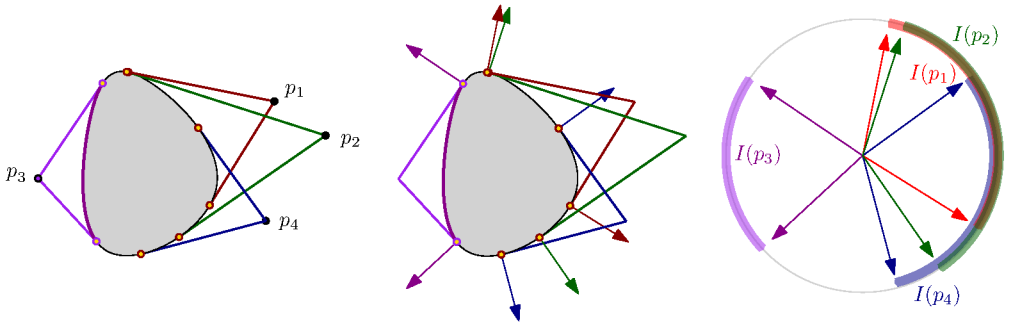


Figure 2.2 Four points and a convex body with their associated circular intervals.

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 shrink 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 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.

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 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.*

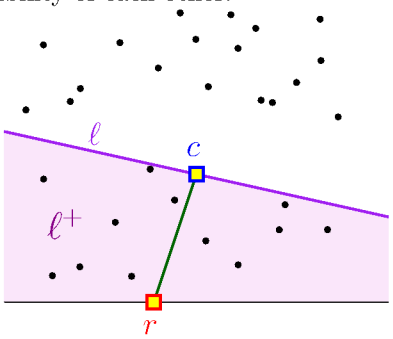
Proof. Let J be the largest independent set of intervals in G. The intervals of J divide the circle into $2|J|$ (atomic) circular arcs. Consider such an arc γ, and let $K(\gamma)$ be the set of all intervals of $\mathcal{V}$ that are fully contained in γ. All the intervals of $K(\gamma)$ are pairwise 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 $|K(\gamma)| \leq \omega$.

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, $|K'(\gamma)| \leq 3\omega$. 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 $\leq 2\alpha$ 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 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 $\mathsf{c}r$ intersects at least $n^2/36$ segments of S.*

Proof. If the segment $\mathsf{c}r$ 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 $\mathsf{c} \in \ell$. Rotate ℓ around c until there are $\geq n/6$ points on each side of $r\mathsf{c}$ in the halfspace ℓ^+. To see why this rotation of ℓ exists, observe that the two halfspaces bounded by the line spanning $r\mathsf{c}$, 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 $\mathsf{c}r$ cannot see each other, as the segment connecting them must intersect $\mathsf{c}r$. Consequently, the number of induced segments that $\mathsf{c}r$ intersects is at least $n^2/36$. ◀

▶ **Lemma 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 $\hbar$ between a centerpoint c of U^+ and the inner approximation. For the halfspace $\hbar^+$ containing c, we have $t_i = |U^+ \cap \hbar^+| \geq |U^+|/3 \geq \omega/3$. Furthermore, all the points of U^+ are pairwise mutually visible (in relation to the inner approximation B_i). Namely, $m_{i+1} = |E(G_i - (U^+ \cap \hbar^+))| \leq m_i - \binom{t_i}{2} \leq m_i - \omega^2/36$.

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 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 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 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 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 $\operatorname{depth}(G_i) \leq t$. 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 $m = m_i = |E(G_i)| = O(\bigcirc_P t^2)$, by Lemma 4, since $\bigcirc_P$ is an upper bound on the size of the largest independent set in G_i. By Lemma 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 $O(m_i/t^2) = O(\bigcirc_P)$ 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 [17] we show that the greedy classification algorithm can be implemented efficiently.

▶ **Lemma 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 $O\big(n \log^2 n \log\log n + T \cdot \bigcirc_P \log n\big)$ expected time.*

Proof. The algorithm follows the proof of Theorem 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 [17].

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 [4] 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 $O(\log n)$ [19, 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 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:

1. there are $\sigma = O(n)$ updates to $\mathcal{T}$, and
2. 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 $O(\sigma \log n \log\log n + n \log n + T \cdot \bigcirc_P)$. As there are $O(\log n)$ epochs, the algorithm can be implemented in expected time $O(n \log^2 n \log\log n + T \cdot \bigcirc_P \log n)$.

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 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 $O(t \log t)$. ◀

In Section 4 we present an application of the greedy classification algorithm. Namely, we present an efficient algorithm for computing the discrete geometric median of a point set (Lemma 18).

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 $\boldsymbol{\epsilon}_i$ 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.

The idea behind the analysis is similar to Theorem 8. 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. Thus we have to analyze mutual visibility between *triples* of points. The analysis considers both the intersection graph G_i between pairs of points, and a new hypergraph H_i, where there is an edge $\{p, q, r\}$ in H_i if the triangle in $\mathbb{R}^3$ formed by the points p, q, r avoids the inner approximation B_i. The main technical ingredient involves bounding the number of edges in H_i by the maximum depth and size of the largest independent set in G_i. Finally, we argue that in each iteration a constant number of edges are deleted from H_i by the centerpoint property. The full details are presented in [12]. We obtain the following result.

▶ **Theorem 10** (Proof in [12]). *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 $P \setminus C$.*

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.

▶ **Lemma 11.** *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 $|Q_{\text{in}}| \geq |\mathcal{CH}(Q_{\text{in}})| \geq |F_{\text{in}}|$.

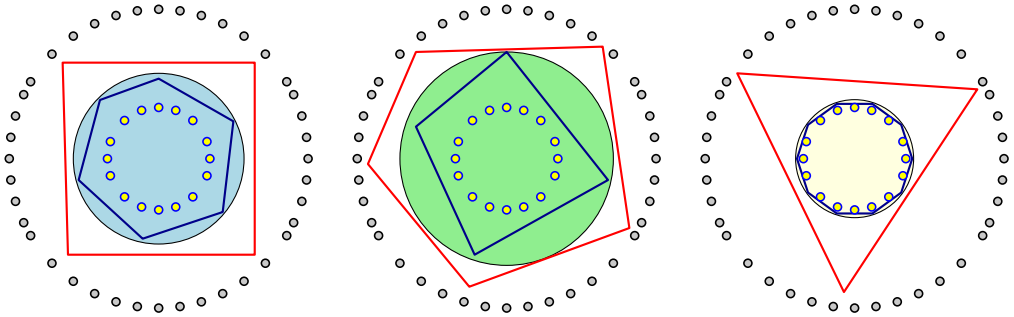


Figure 3.1 The separation price, for the same point set, is different depending on how "tight" the body is in relation to the inner and outer point set.

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. ◀

▶ Remarks.

1. Naturally the separation price, and thus the proof of the lower bound, generalizes to higher dimensions. See [12].
2. The lower bound only holds for $d \geq 2$. In 1D, the problem can be solved using $O(\log n)$ queries with binary search. The above would predict that any algorithm needs $\Omega(1)$ queries. However it is not hard to argue a stronger lower bound of $\Omega(\log n)$.
3. In [12], 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.

3.2 A sketch of the improved algorithm

We refer to reader to [12] for a complete description of the improved algorithm in 2D. The idea of the algorithm is conceptually the same as the greedy algorithm of Section 2: at all times a current inner approximation $B \subseteq C$ and the set of unclassified points $U \subseteq P$ are maintained. We define a *pocket* to be a connected region of $\mathcal{CH}(U \cup B) \setminus B$ (see Figure 3.2). The algorithm will repeatedly choose points inside a pocket, and attempt to classify them, while simultaneously dividing the pocket into two smaller pockets. In this way, we improve the inner approximation B every time a pocket is handled. The algorithm continues in this fashion until all points are classified. The analysis involves a careful charging argument. Roughly speaking, whenever a pocket contains a vertex of F_{in} or F_{out}, we can charge the work of creating and splitting the pocket to such a vertex. Otherwise, a pocket contains no vertex from either fence. For such a pocket, we prove that when this pocket was created by the algorithm, all of the points contained in the pocket must lie outside C. When all points inside a pocket are outside C, we argue that they can all be classified as outside after $O(\log n)$ queries by using centerpoints as the oracle queries.

▶ **Theorem 12** (Proof in [12]). *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 $P \setminus C$.*

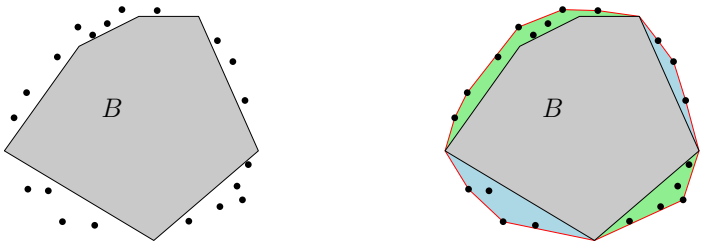


Figure 3.2 Unclassified points and their pockets.

4 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 $O(\bigcirc_P \log^2 n)$ evaluations to f or its derivative.

▶ **Definition 13.** *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 $\mathcal{L}_f(c) = \left\{p \in \mathbb{R}^d \;\middle|\; f(p) \leq c\right\}$. If f is a convex function, then $\mathcal{L}_f(c)$ is a convex set for all $c \in \mathbb{R}$.*

▶ **Definition 14.** *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 $q \in \mathbb{R}^d$, $f(q) \geq f(p) + \langle v, q - p\rangle$. 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 for 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 [8, 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 15.** *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 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 $c \leq r$, 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. 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 8.

The implementation details of Theorem 8 are given in Lemma 9. ◀

The algorithm

Let $\alpha = \min_{p \in P} f(p)$. For a given number $r \geq 0$, 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 15. In doing so, we modify the separation oracle of Lemma 15 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 $|P_{j+1}| \leq 1$. 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 15 only a logarithmic number of times.

▶ **Lemma 16.** *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 $V_r = \{i \in V \mid i \leq r\}$. 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 $\left|V_{r_j}\right| \leq 1$.

We can assume without loss of generality that $V = \{1, 2, \ldots, N\}$. For an integer $i \leq N$, 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 17.** *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 $O(\bigcirc_P \log^2 n)$ 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 $O(n \log^3 n \log\log n + T \cdot \bigcirc_P \log^2 n)$ expected time.*

Proof. The result follows by combining Lemma 15 and Lemma 16. ◀

4.1 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 17, we obtain the following.

▶ **Lemma 18.** *Let P be a set of points in $\mathbb{R}^2$. Then the discrete geometric median of P can be computed in $O(n \log^2 n \cdot (\log n \log\log n + \bigcirc_P))$ expected time.*

▶ Remark. For a set of n points P chosen uniformly at random from the unit square, it is known that in expectation $\bigcirc_P = \Theta(n^{1/3})$ [1]. 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.

5 Conclusion and open problems

In this paper we have presented various algorithms for classifying points with oracle access to an unknown convex body. As far as the authors are aware, this problem has not been studied within the community previously. However we believe that this is an interesting and natural problem. We now pose some open problems.

(A) Develop a more natural instance-optimal algorithm in 2D which improves upon the $O(\log^2 n)$ approximation. Alternatively, develop algorithms in which the number of queries is parameterized by different functions of the input instance.

(B) An algorithm in 3D which is instance-optimal up to some additional factors (see [12] for the definition of the separation price in higher dimensions).

(C) Any results beyond three dimensions is unknown. The greedy algorithm (Theorem 8 and Theorem 10) easily extends to $\mathbb{R}^d$. However the analysis in higher dimensions will most likely reveal that the algorithm makes (ignoring logarithmic factors) of the order of ${\bigcirc_P}^{O(d)}$ queries, which is only interesting when $\bigcirc_P$ is much smaller than n.

References

1 G. Ambrus and I. Bárány. Longest convex chains. *Rand. Struct. & Alg.*, 35(2):137–162, 2009. doi:10.1002/rsa.20269.

2 Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987. doi:10.1007/BF00116828.

3 Imre Bárány and Zoltán Füredi. Computing the volume is difficulte. *Discrete Comput. Geom.*, 2:319–326, 1987. doi:10.1007/BF02187886.

4 T. M. Chan. An optimal randomized algorithm for maximum Tukey depth. In J. Ian Munro, editor, *Proc. 15th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 430–436. SIAM, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982853.

5 Michael B. Cohen, Yin Tat Lee, Gary L. Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In Daniel Wichs and Yishay Mansour, editors, *Proc. 48th ACM Sympos. Theory Comput.* (STOC), pages 9–21. ACM, 2016. doi:10.1145/2897518.2897647.

6 David A. Cohn, Les E. Atlas, and Richard E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994. doi:10.1007/BF00993277.

7 Esther Ezra and Micha Sharir. A nearly quadratic bound for point-location in hyperplane arrangements, in the linear decision tree model. *Discrete Comput. Geom.*, 61(4):735–755, 2019. doi:10.1007/s00454-018-0043-8.

8 Juan Ferrera. *An introduction to nonsmooth analysis*. Academic Press, Boston, 2013. doi:10.1016/C2013-0-15234-8.

9 Yuhong Guo and Russell Greiner. Optimistic active-learning using mutual information. In *Proc. 20th Int. Joint Conf. on AI* (IJCAI), pages 823–829, 2007. URL: http://ijcai.org/Proceedings/07/Papers/132.pdf.

10 S. Har-Peled, M. Jones, and S. Rahul. An animation of the greedy classification algorithm in 2d. URL: https://www.youtube.com/watch?v=IZX0VQdIgNA.

11 S. Har-Peled, N. Kumar, D. M. Mount, and B. Raichel. Space exploration via proximity search. *Discrete Comput. Geom.*, 56(2):357–376, 2016. doi:10.1007/s00454-016-9801-7.

12 Sariel Har-Peled, Mitchell Jones, and Saladi Rahul. Active learning a convex body in low dimensions. *CoRR*, abs/1903.03693, 2019. arXiv:1903.03693.

13 David Haussler and Emo Welzl. ε-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987. doi:10.1007/BF02187876.

14 Daniel M. Kane, Shachar Lovett, Shay Moran, and Jiapeng Zhang. Active classification with comparison queries. In *Proc. 58th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 355–366, 2017. doi:10.1109/FOCS.2017.40.

15 Andrey Kupavskii. The vc-dimension of k-vertex d-polytopes. *CoRR*, abs/2004.04841, 2020. arXiv:2004.04841.

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

17 Kurt Mehlhorn and Stefan Näher. Dynamic fractional cascading. *Algorithmica*, 5(2):215–241, 1990. doi:10.1007/BF01840386.

18 F. Panahi, A. Adler, A. F. van der Stappen, and K. Goldberg. An efficient proximity probing algorithm for metrology. In *Int. Conf. on Automation Science and Engineering, CASE 2013*, pages 342–349, 2013. doi:10.1109/CoASE.2013.6653995.

19 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction*. Texts and Monographs in Computer Science. Springer, 1985. doi:10.1007/978-1-4612-1098-6.

20 Natan Rubin. An improved bound for weak epsilon-nets in the plane. In Mikkel Thorup, editor, *Proc. 59th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 224–235. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00030.

21 Burr Settles. Active learning literature survey. Technical Report #1648, Computer Science, Univ. Wisconsin, Madison, January 2009. URL: https://minds.wisconsin.edu/bitstream/handle/1793/60660/TR1648.pdf?sequence=1&isAllowed=y.

22 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.

23 Wolfgang Weil, editor. *Random Polytopes, Convex Bodies, and Approximation*, pages 77–118. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-38175-4_2.