Active Learning a Convex Body in Low Dimensions

Sariel Har-Peled, Mitchell Jones and Rahul Saladi

UIUC Theory Seminar, November 11, 2019

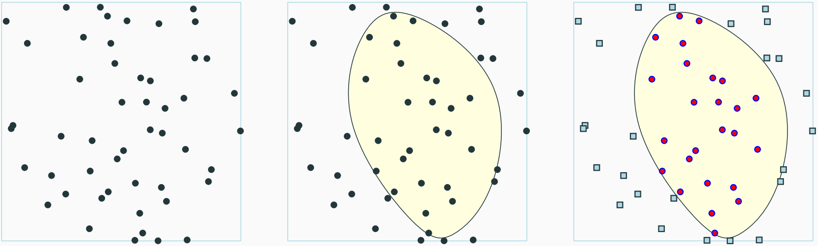
An innocent problem

Problem

Input: $P \subset \mathbb{R}^2$, oracle for unknown convex body C.

Oracle: Query $q \in \mathbb{R}^2$, returns true $\iff q \in C$.

Goal: Compute $P \cap C$ using fewest number of oracle queries.

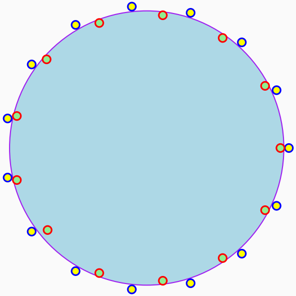


Motivation: Active learning

- Input space X
- Learner data: $x_1, \ldots, x_n \in X$ (without labels)
- Learner can query oracle for label of any $q \in X$
- Build classifier using few queries
- What queries to choose?

Bad news

- Worst case: query all points
- **Question:** More interesting model to study?

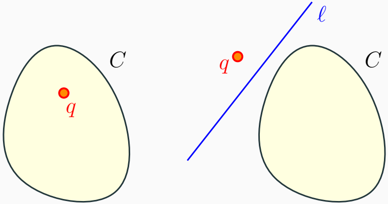


Modified problem

Problem

Input: $P \subset \mathbb{R}^2$, oracle for unknown convex body C.

Oracle: Separation oracle



Goal: Compute $P \cap C$ using fewest number of oracle queries.

Motivation

- Slightly stronger model

Motivation

- Slightly stronger model
- Separation oracles are well-known (OR)

Motivation

- Slightly stronger model
- Separation oracles are well-known (OR)
- Computational problems with oracle access:

Motivation

- Slightly stronger model
- Separation oracles are well-known (OR)
- Computational problems with oracle access:
    - Nearest-neighbor oracles [Har-Peled, Kumar, et al., 2016]

Motivation

- Slightly stronger model
- Separation oracles are well-known (OR)
- Computational problems with oracle access:
  - Nearest-neighbor oracles [Har-Peled, Kumar, et al., 2016]
  - Proximity probe [Panahi, Adler, et al., 2013]

Motivation

- Slightly stronger model
- Separation oracles are well-known (OR)
- Computational problems with oracle access:
  - Nearest-neighbor oracles [Har-Peled, Kumar, et al., 2016]
  - Proximity probe [Panahi, Adler, et al., 2013]
- Minimizing communication complexity

One approach: PAC learning

- Allow error in classification

One approach: PAC learning

- Allow error in classification
- Algorithm:

One approach: PAC learning

- Allow error in classification
- Algorithm:
    1. Randomly sample input

One approach: PAC learning

- Allow error in classification
- Algorithm:
    1. Randomly sample input
    2. Obtain labels for sample

One approach: PAC learning

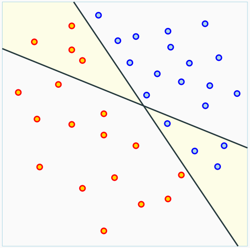
- Allow error in classification
- Algorithm:
    1. Randomly sample input
    2. Obtain labels for sample
    3. Classify sample

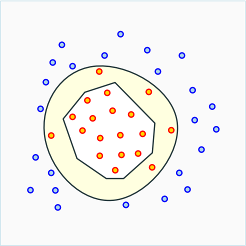
One approach: PAC learning

- Allow error in classification
- Algorithm:
  1. Randomly sample input
  2. Obtain labels for sample
  3. Classify sample
- Size of sample?

One approach: PAC learning

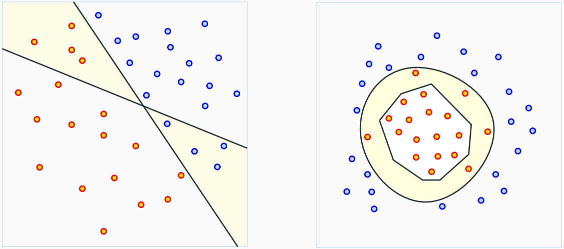
- Misclassified points = symmetric difference of learned and true classifier





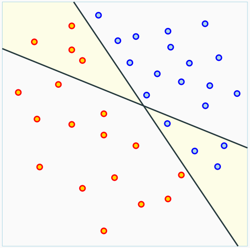
One approach: PAC learning

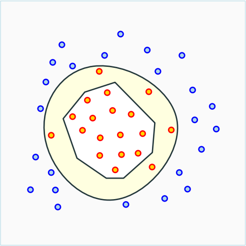
- Misclassified points = symmetric difference of learned and true classifier
- Halfplane $\implies$ symmetric difference is a wedge



One approach: PAC learning

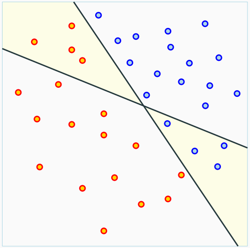
- Misclassified points = symmetric difference of learned and true classifier
- Halfplane $\implies$ symmetric difference is a wedge
- Wedge has finite VC dimension $\implies$ random sample of size $\approx O(\varepsilon^{-1} \log \varepsilon^{-1}) \implies \varepsilon n$ error

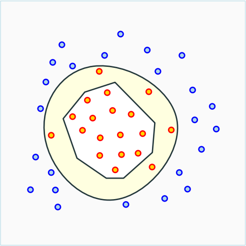




One approach: PAC learning

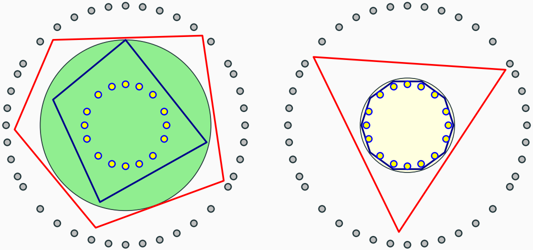
- Misclassified points = symmetric difference of learned and true classifier
- Halfplane $\implies$ symmetric difference is a wedge
- Wedge has finite VC dimension $\implies$ random sample of size $\approx O(\varepsilon^{-1}\log\varepsilon^{-1}) \implies \varepsilon n$ error
- Scheme fails for arbitrary convex regions





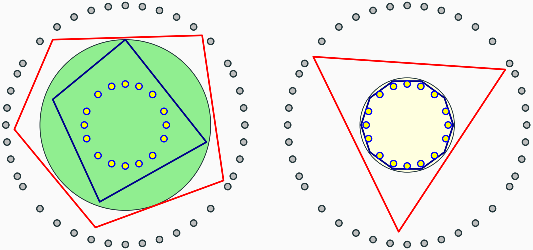
Hard vs. easy instances

- Worst case: query all points

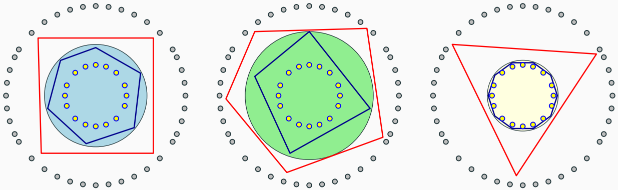


Hard vs. easy instances

- Worst case: query all points
- **Goal:** design instance sensitive algorithms

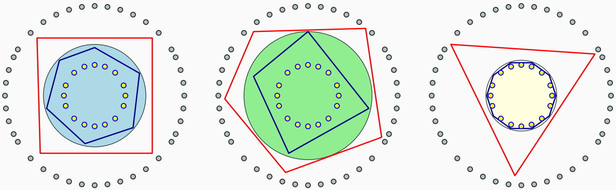


A lower bound



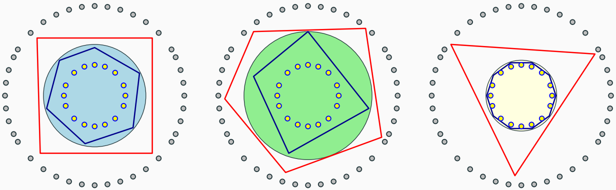
- F_{in} = convex polygon with fewest vertices s.t. $F_{\text{in}} \subseteq C$ and $C \cap P = F_{\text{in}} \cap P$.

A lower bound



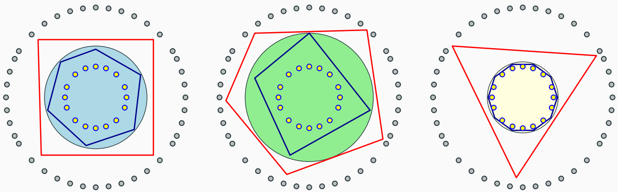
- F_{in} = convex polygon with fewest vertices s.t. $F_{\mathrm{in}} \subseteq C$ and $C \cap P = F_{\mathrm{in}} \cap P$.
- F_{out} = convex polygon with fewest vertices s.t. $C \subseteq F_{\mathrm{out}}$ and $C \cap P = F_{\mathrm{out}} \cap P$.

A lower bound



- F_{in} = convex polygon with fewest vertices s.t. $F_{\text{in}} \subseteq C$ and $C \cap P = F_{\text{in}} \cap P$.
- F_{out} = convex polygon with fewest vertices s.t. $C \subseteq F_{\text{out}}$ and $C \cap P = F_{\text{out}} \cap P$.
- Separation price $\sigma(P, C) = |F_{\text{in}}| + |F_{\text{out}}|$.

A lower bound

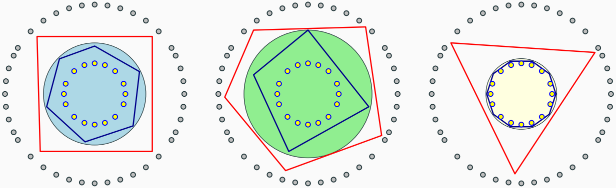


- F_{in} = convex polygon with fewest vertices s.t. $F_{\text{in}} \subseteq C$ and $C \cap P = F_{\text{in}} \cap P$.
- F_{out} = convex polygon with fewest vertices s.t. $C \subseteq F_{\text{out}}$ and $C \cap P = F_{\text{out}} \cap P$.
- Separation price $\sigma(P, C) = |F_{\text{in}}| + |F_{\text{out}}|$.

Lemma

Any algorithm must make at least $\sigma(P, C)$ oracle queries.

A lower bound

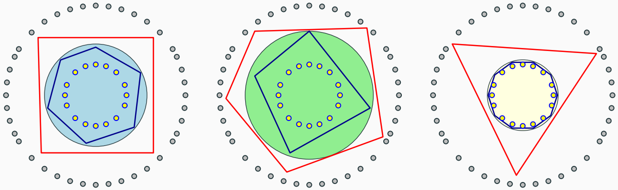


Lemma

Any algorithm must make at least $\sigma(P, C)$ oracle queries.

Proof.

A lower bound

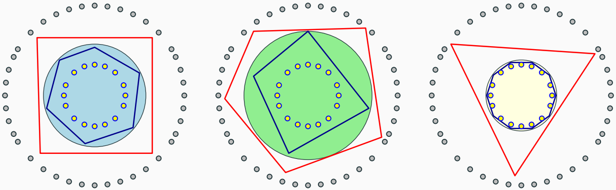


Lemma

Any algorithm must make at least $\sigma(P, C)$ oracle queries.

Proof.

- Q: set of queries, $Q_{\text{in}} = C \cap Q$, $K = \text{CH}(Q_{\text{in}})$



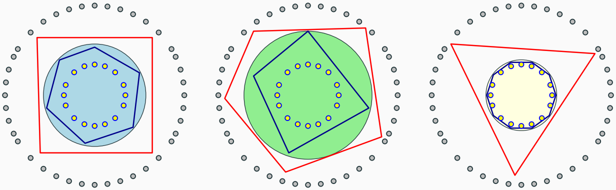
Lemma

Any algorithm must make at least $\sigma(P, C)$ oracle queries.

Proof.

- Q: set of queries, $Q_{\mathrm{in}} = C \cap Q$, $K = \mathrm{CH}(Q_{\mathrm{in}})$
- $K \subseteq C$ and $K \cap P = C \cap P$

A lower bound



Lemma

Any algorithm must make at least $\sigma(P, C)$ oracle queries.

Proof.

- Q: set of queries, $Q_{\text{in}} = C \cap Q$, $K = \mathrm{CH}(Q_{\text{in}})$
- $K \subseteq C$ and $K \cap P = C \cap P$

  $\implies |Q_{\text{in}}| \geqslant |K| \geqslant |F_{\text{in}}|$ □

Results

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P) \log n)$ (†)

(†) $k(P) =$ largest # of pts of P in convex position

Results

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P) \log n)$ (†)
Classify (2D)	$\sigma(P, C)$	$O(\sigma(P, C) \log^2 n)$

(†) $k(P) =$ largest # of pts of P in convex position

Results

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P) \log n)$ (†)
Classify (2D)	$\sigma(P, C)$	$O(\sigma(P, C) \log^2 n)$
Classify (3D)	—	$O(k(P) \log n)$ (†)

(†) $k(P) =$ largest # of pts of P in convex position

Results

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P) \log n)$ ($\dagger$)
Classify (2D)	$\sigma(P, C)$	$O(\sigma(P, C) \log^2 n)$
Classify (3D)	—	$O(k(P) \log n)$ ($\dagger$)
Verify in (2D)	$\lvert F_{\mathrm{in}} \rvert$	$O(\lvert F_{\mathrm{in}} \rvert \log n)$

($\dagger$) $k(P) =$ largest # of pts of P in convex position

Results

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P) \log n)$ (†)
Classify (2D)	$\sigma(P, C)$	$O(\sigma(P, C) \log^2 n)$
Classify (3D)	—	$O(k(P) \log n)$ (†)
Verify in (2D)	$\|F_{\mathrm{in}}\|$	$O(\|F_{\mathrm{in}}\| \log n)$
Verify out (2D)	$\|F_{\mathrm{out}}\|$	$O(\|F_{\mathrm{out}}\| \log n)$ (‡)

(†) $k(P) =$ largest # of pts of P in convex position

(‡) Randomized, w.h.p

Results

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P) \log n)$ (†)
Classify (2D)	$\sigma(P, C)$	$O(\sigma(P, C) \log^2 n)$
Classify (3D)	—	$O(k(P) \log n)$ (†)
Verify in (2D)	$\lvert F_{\mathrm{in}} \rvert$	$O(\lvert F_{\mathrm{in}} \rvert \log n)$
Verify out (2D)	$\lvert F_{\mathrm{out}} \rvert$	$O(\lvert F_{\mathrm{out}} \rvert \log n)$ (‡)

(†) $k(P) =$ largest # of pts of P in convex position

(‡) Randomized, w.h.p

First attempt: A greedy algorithm

The greedy algorithm: preliminaries

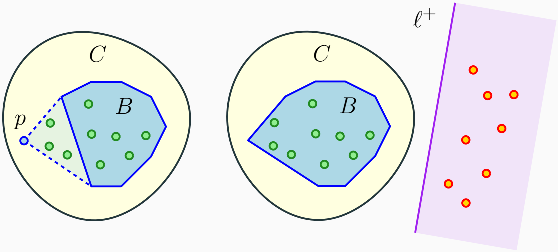
- Maintain approximation $B \subseteq C$

The greedy algorithm: preliminaries

- Maintain approximation $B \subseteq C$
- Operations:

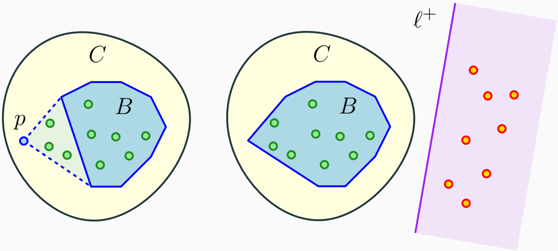
The greedy algorithm: preliminaries

- Maintain approximation $B \subseteq C$
- Operations:
  1. **expand**(p): Update $B = \mathrm{CH}(B + p)$
  2. **remove**(ℓ^+): Classify points $P \cap \ell^+$ as outside C



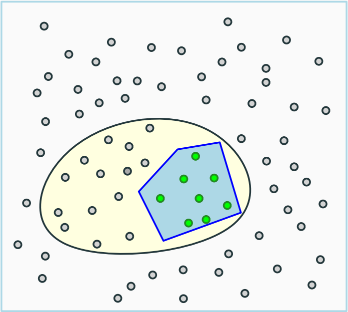
The greedy algorithm: preliminaries

- Maintain approximation $B \subseteq C$
- Operations:
  1. **expand**(p): Update $B = \mathrm{CH}(B + p)$
  2. **remove**(ℓ^+): Classify points $P \cap \ell^+$ as outside C
- $c \in \mathbb{R}^2$ is a centerpoint for P if for all halfspaces ℓ^+:
  $c \in \ell^+ \implies |P \cap \ell^+| \geqslant |P|/3$.



The greedy algorithm

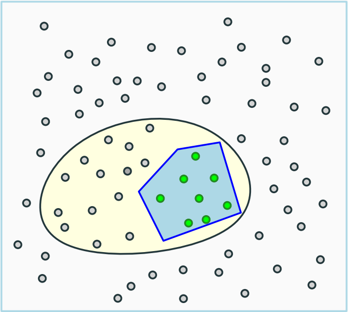
$U \subseteq P$ unclassified points. While $U \neq \varnothing$:



The greedy algorithm

$U \subseteq P$ unclassified points. While $U \neq \varnothing$:

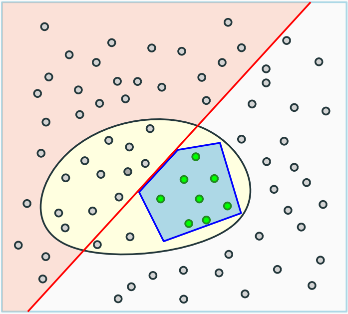
1. $\ell^+ =$ halfspace tangent to B maximizing $|\ell^+ \cap U|$



The greedy algorithm

$U \subseteq P$ unclassified points. While $U \neq \varnothing$:

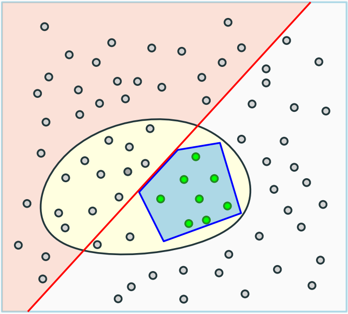
1. $\ell^+ =$ halfspace tangent to B maximizing $|\ell^+ \cap U|$



The greedy algorithm

$U \subseteq P$ unclassified points. While $U \neq \varnothing$:

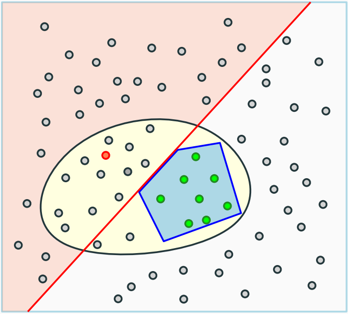
1. $\ell^+ =$ halfspace tangent to B maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$



The greedy algorithm

$U \subseteq P$ unclassified points. While $U \neq \varnothing$:

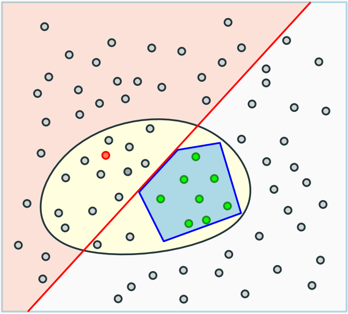
1. $\ell^+ =$ halfspace tangent to B maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$



The greedy algorithm

$U \subseteq P$ unclassified points. While $U \neq \varnothing$:

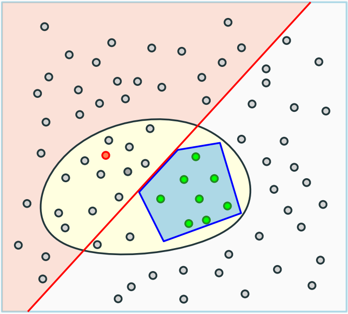
1. $\ell^+ =$ halfspace tangent to B maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$
3. Query oracle using c:



The greedy algorithm

$U \subseteq P$ unclassified points. While $U \neq \varnothing$:

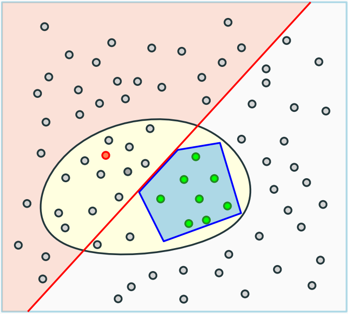
1. $\ell^+ =$ halfspace tangent to B maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$
3. Query oracle using c:
   (A) $c \in C \implies$ **expand**(c)



The greedy algorithm

$U \subseteq P$ unclassified points. While $U \neq \varnothing$:

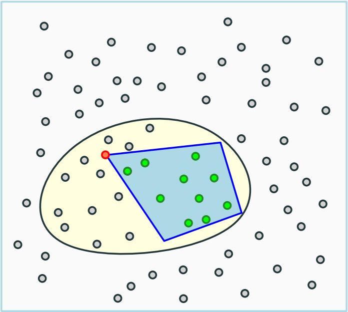
1. $\ell^+ =$ halfspace tangent to B maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$
3. Query oracle using c:
   (A) $c \in C \implies$ **expand**(c)
   (B) $c \notin C$, h is a separating line $\implies$ **remove**(h)



The greedy algorithm

$U \subseteq P$ unclassified points. While $U \neq \varnothing$:

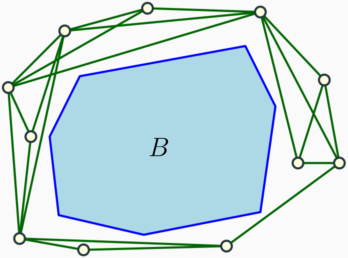
1. $\ell^+ =$ halfspace tangent to B maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$
3. Query oracle using c:
   (A) $c \in C \implies$ **expand**(c)
   (B) $c \notin C$, h is a separating line $\implies$ **remove**(h)



Animation

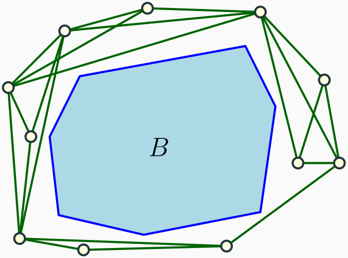
Analysis

- Count visible pairs of points



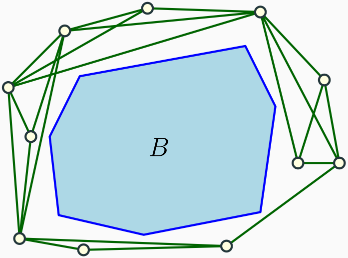
Analysis

- Count visible pairs of points
- In each iteration:



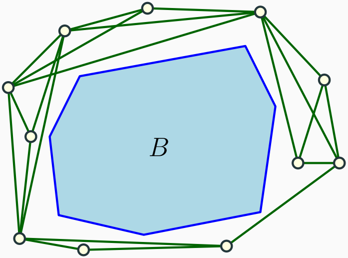
Analysis

- Count visible pairs of points
- In each iteration:
  - (A) Pairs lose visibility



Analysis

- Count visible pairs of points
- In each iteration:
  - (A) Pairs lose visibility
  - (B) Classify points

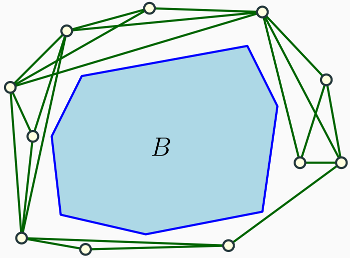


Analysis

- Count visible pairs of points
- In each iteration:
  - (A) Pairs lose visibility
  - (B) Classify points

Lemma

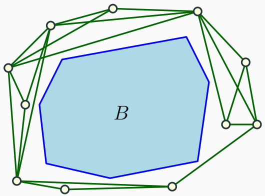
Number of visible pairs decrease by a (roughly) constant fraction in each iteration.



Two interpretations of the visibility graph

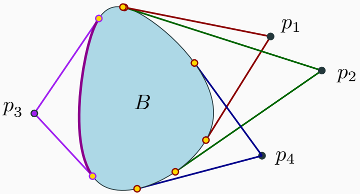
Visibility graph $G_B = (P, E)$:

$(p, q) \in E \iff pq \cap B = \varnothing$



$p \in P$ has interval $I(p)$

$(p, q) \in E \iff I(p) \cap I(q) \neq \varnothing$

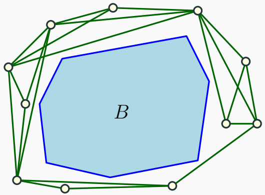


Two observations

Observations:

1. $Q \subseteq P$ independent set in $G_B \implies Q$ is in convex position

$$(p, q) \in E \iff pq \cap B = \emptyset$$

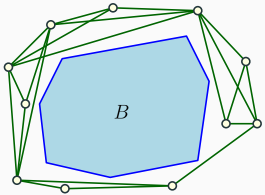


Two observations

Observations:

1. $Q \subseteq P$ independent set in $G_B \implies Q$ is in convex position
2. $Q \subseteq P$ and B are linearly separable $\implies Q$ clique in G_B

$$(p, q) \in E \iff pq \cap B = \varnothing$$

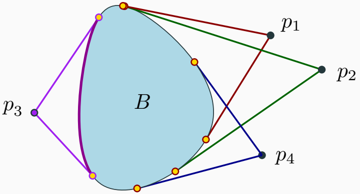


Lemma 1

$\alpha(G_B)$ = size of largest indep. set, $\omega(G_B)$ = maximum depth, then $|E| = O(\alpha(G_B)\omega(G_B)^2)$.

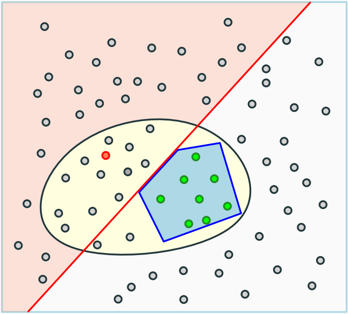
$p \in P$ has interval $I(p)$

$(p, q) \in E \iff I(p) \cap I(q) \neq \varnothing$



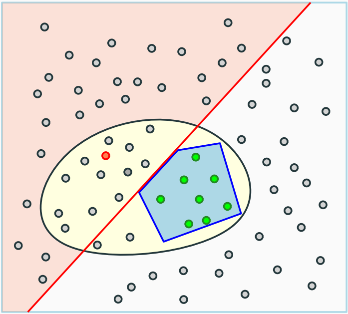
How many edges are removed?

- Compute halfspace ℓ^+ tangent to B maximizing $|\ell^+ \cap U|$



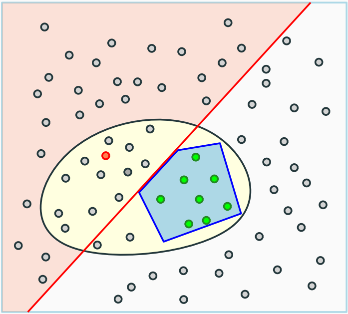
How many edges are removed?

- Compute halfspace ℓ^+ tangent to B maximizing $|\ell^+ \cap U|$
- Set $m = |\ell^+ \cap U| \geqslant \omega(G_B)$



How many edges are removed?

- Compute halfspace ℓ^+ tangent to B maximizing $|\ell^+ \cap U|$
- Set $m = |\ell^+ \cap U| \geqslant \omega(G_B)$
- c = centerpoint of $\ell^+ \cap U$

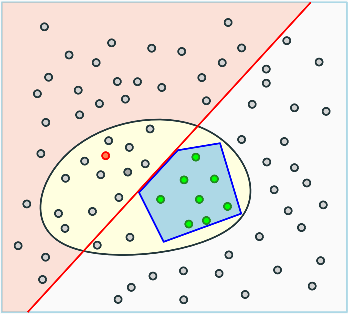


How many edges are removed?

- Compute halfspace ℓ^+ tangent to B maximizing $|\ell^+ \cap U|$
- Set $m = |\ell^+ \cap U| \geqslant \omega(G_B)$
- c = centerpoint of $\ell^+ \cap U$

Lemma 2

When $c \in C$, **expand**(c) deletes $\geqslant \omega(G_B)^2/36$ edges from G_B.



Putting it all together

Lemma 1

$\alpha(G_B)$ = size of largest indep. set, $\omega(G_B)$ = maximum depth, then $|E| = O(\alpha(G_B)\omega(G_B)^2)$.

Lemma 2

When $c \in C$, **expand**(c) deletes $\geqslant \omega(G_B)^2/36$ edges from G_B.

Putting it all together

Lemma 1

$\alpha(G_B)$ = size of largest indep. set, $\omega(G_B)$ = maximum depth, then $|E| = O(\alpha(G_B)\omega(G_B)^2)$.

Lemma 2

When $\mathscr{c} \in C$, **expand**$(\mathscr{c})$ deletes $\geqslant \omega(G_B)^2/36$ edges from G_B.

Our result

Greedy algorithm classifies all points using $O(k(P) \log n)$ queries.

Putting it all together

Lemma 1

$\alpha(G_B) =$ size of largest indep. set, $\omega(G_B) =$ maximum depth, then $|E| = O(\alpha(G_B)\omega(G_B)^2) = O(k(P)\omega(G_B)^2)$.

Lemma 2

When $c \in C$, **expand**(c) deletes $\geqslant \omega(G_B)^2/36$ edges from G_B.

Our result

Greedy algorithm classifies all points using $O(k(P) \log n)$ queries.

Extending the algorithm to 3D

Extending the algorithm

$U \subseteq P$ unclassified points. While $U \neq \varnothing$:

1. $\ell^+ =$ halfspace tangent to B maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$
3. Query oracle using c:
   (A) $c \in C \implies$ **expand**(c)
   (B) $c \notin C$, h is a separating line $\implies$ **remove**(h)

Extending the algorithm

$U \subseteq P$ unclassified points. While $U \neq \varnothing$:

1. $\ell^+ =$ halfspace tangent to B maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$
3. Query oracle using c:
   (A) $c \in C \implies$ **expand**(c)
   (B) $c \notin C$, h is a separating plane $\implies$ **remove**(h)

Extending the analysis

- When B is expanded, pairs of points do not lose visibility!

Extending the analysis

- When B is expanded, pairs of points do not lose visibility!
- Need to consider triples of points

Extending the analysis

- When B is expanded, pairs of points do not lose visibility!
- Need to consider triples of points
- Maintain two graphs (w.r.t B):

Extending the analysis

- When B is expanded, pairs of points do not lose visibility!
- Need to consider triples of points
- Maintain two graphs (w.r.t B):
    1. $G_B = (P, E)$, $(p, q) \in E \iff pq$ avoids B

Extending the analysis

- When B is expanded, pairs of points do not lose visibility!
- Need to consider triples of points
- Maintain two graphs (w.r.t B):
    1. $G_B = (P, E)$, $(p, q) \in E \iff pq$ avoids B
    2. Hypergraph $H_B = (P, \mathcal{E})$, $\{p, q, r\} \in \mathcal{E} \iff$ triangle pqr avoids B

Everything still works

Lemma 1

$\alpha(G_B) =$ size of largest indep. set, $\omega =$ maximum depth, then $|\mathcal{E}(H_B)| = \Theta(\alpha(G_B)\omega^3)$.

Lemma 2

When $\mathcal{c} \in C$, **expand**$(\mathcal{c})$ deletes $\geqslant \omega(G_B)^3/c$ triangles from H_B.

Everything still works

Lemma 1

$\alpha(G_B)$ = size of largest indep. set, ω = maximum depth, then $|\mathcal{E}(H_B)| = \Theta(\alpha(G_B)\omega^3)$.

Lemma 2

When $c \in C$, **expand**(c) deletes $\geqslant \omega(G_B)^3/c$ triangles from H_B.

Our result

Greedy algorithm classifies all points using $O(k(P)\log n)$ queries.

An instance optimal algorithm in 2D

Preliminaries

- Maintain inner approximation $B \subseteq C$

Preliminaries

- Maintain inner approximation $B \subseteq C$
- Query is more carefully chosen

Preliminaries

- Maintain inner approximation $B \subseteq C$
- Query is more carefully chosen
- Two operations:

Preliminaries

- Maintain inner approximation $B \subseteq C$
- Query is more carefully chosen
- Two operations:
    1. Directional climb

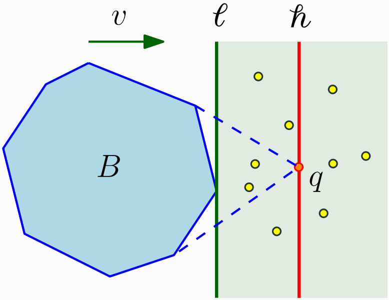
Preliminaries

- Maintain inner approximation $B \subseteq C$
- Query is more carefully chosen
- Two operations:
    1. Directional climb
    2. Pocket splitting

Directional climbs

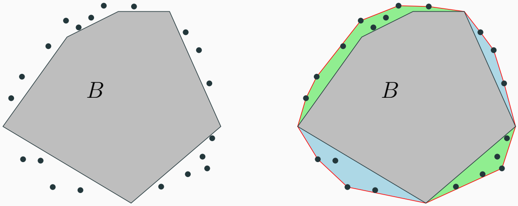
Given direction v:

- Compute line ℓ tangent to B, perpendicular to v
- Regular iteration on $\ell^+ \cap U$.



Pockets

Pocket: A connected region of $\mathrm{CH}(U \cup B) \setminus B$



Lemma

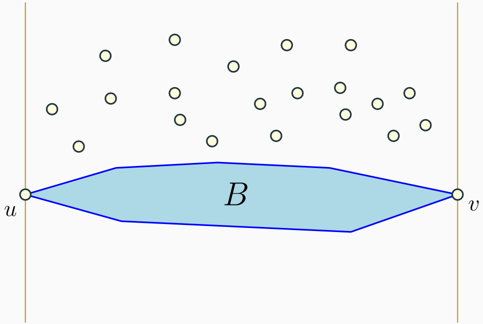
In $O(\log n)$ oracle queries, can split a pocket Υ, into two pockets Υ_1, Υ_2, $|\Upsilon_i \cap P| \leqslant (2/3)|\Upsilon \cap P|$.

Algorithm sketch

1. Vertical climb in positive & negative direction of x-axis

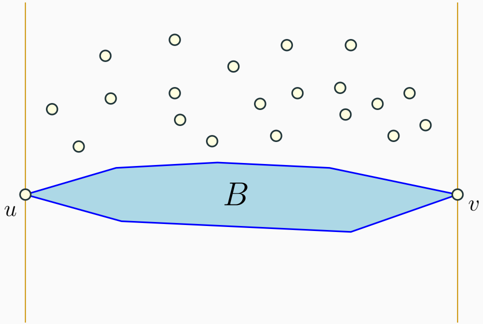
Algorithm sketch

1. Vertical climb in positive & negative direction of x-axis



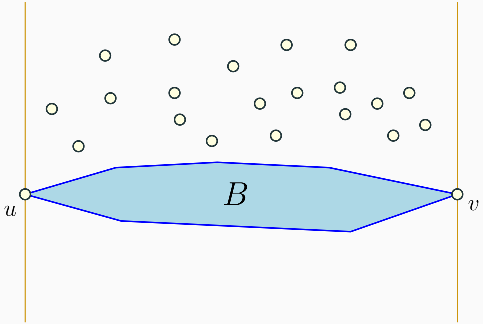
Algorithm sketch

1. Vertical climb in positive & negative direction of x-axis
2. Obtain a segment $uv \subseteq B \subseteq C$



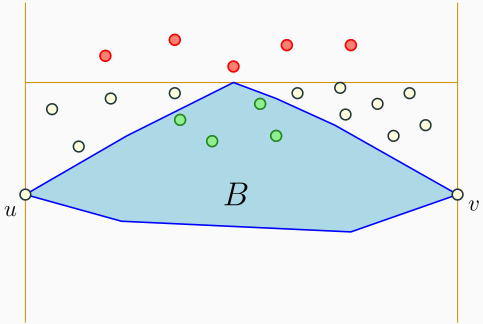
Algorithm sketch

1. Vertical climb in positive & negative direction of x-axis
2. Obtain a segment $uv \subseteq B \subseteq C$
3. Repeatably split non-empty pockets



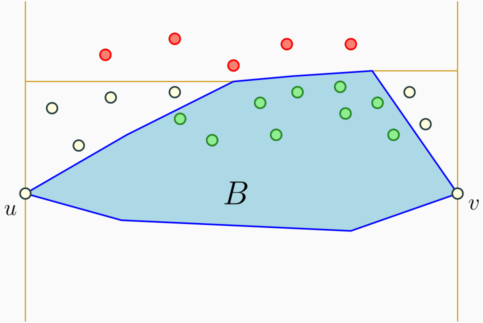
Algorithm sketch

1. Vertical climb in positive & negative direction of x-axis
2. Obtain a segment $uv \subseteq B \subseteq C$
3. Repeatably split non-empty pockets

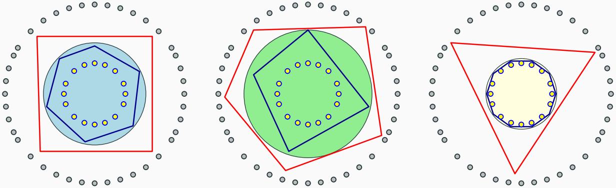


Algorithm sketch

1. Vertical climb in positive & negative direction of x-axis
2. Obtain a segment $uv \subseteq B \subseteq C$
3. Repeatably split non-empty pockets

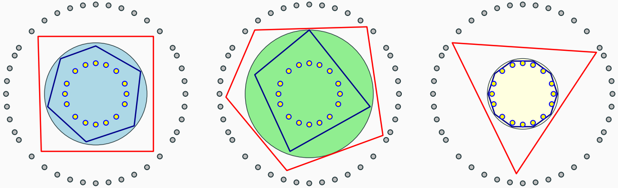


Analysis idea



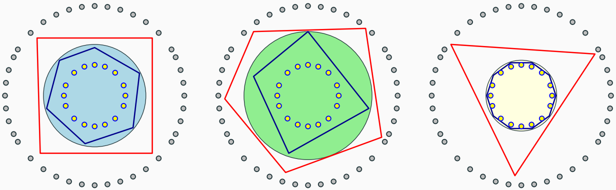
- If a pocket contains a vertex v of inner/outer fence, charge creation and splitting of pocket to v

Analysis idea



- If a pocket contains a vertex v of inner/outer fence, charge creation and splitting of pocket to v
- Else pocket does not contain a vertex of inner/outer fence $\implies$ all points in pocket are outside C

Analysis idea



- If a pocket contains a vertex v of inner/outer fence, charge creation and splitting of pocket to v
- Else pocket does not contain a vertex of inner/outer fence $\implies$ all points in pocket are outside C

Our result

Can classify all points using $O(\sigma(P, C) \log^2 n)$ oracle queries.

Conclusions

Conclusion & open problems

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P) \log n)$ $O(\sigma(P, C) \log^2 n)$
Classify (3D)	—	$O(k(P) \log n)$
Verify in	$\lvert F_{\mathrm{in}} \rvert$	$O(\lvert F_{\mathrm{in}} \rvert \log n)$
Verify out	$\lvert F_{\mathrm{out}} \rvert$	$O(\lvert F_{\mathrm{out}} \rvert \log n)$

Conclusion & open problems

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P)\log n)$ $O(\sigma(P, C)\log^2 n)$
Classify (3D)	—	$O(k(P)\log n)$
Verify in	$\lvert F_{\mathrm{in}}\rvert$	$O(\lvert F_{\mathrm{in}}\rvert\log n)$
Verify out	$\lvert F_{\mathrm{out}}\rvert$	$O(\lvert F_{\mathrm{out}}\rvert\log n)$

- Shaving log factors?

Conclusion & open problems

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P) \log n)$ $O(\sigma(P, C) \log^2 n)$
Classify (3D)	—	$O(k(P) \log n)$
Verify in	$\lvert F_{\text{in}} \rvert$	$O(\lvert F_{\text{in}} \rvert \log n)$
Verify out	$\lvert F_{\text{out}} \rvert$	$O(\lvert F_{\text{out}} \rvert \log n)$

- Shaving log factors?
- Near-optimal solution in 3D?

Conclusion & open problems

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P) \log n)$ $O(\sigma(P, C) \log^2 n)$
Classify (3D)	—	$O(k(P) \log n)$
Verify in	$\lvert F_{\mathrm{in}} \rvert$	$O(\lvert F_{\mathrm{in}} \rvert \log n)$
Verify out	$\lvert F_{\mathrm{out}} \rvert$	$O(\lvert F_{\mathrm{out}} \rvert \log n)$

- Shaving log factors?
- Near-optimal solution in 3D?
- Higher dimensions?

Conclusion & open problems

Problem	Lowerbound	Upperbound
Classify (2D)	$\sigma(P, C)$	$O(k(P) \log n)$ $O(\sigma(P, C) \log^2 n)$
Classify (3D)	—	$O(k(P) \log n)$
Verify in	$\lvert F_{\mathrm{in}} \rvert$	$O(\lvert F_{\mathrm{in}} \rvert \log n)$
Verify out	$\lvert F_{\mathrm{out}} \rvert$	$O(\lvert F_{\mathrm{out}} \rvert \log n)$

- Shaving log factors?
- Near-optimal solution in 3D?
- Higher dimensions?
- Conjecture: Greedy extends to $\mathbb{R}^d$ ($d \geqslant 3$), queries depend exponentially on d

References i

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

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.

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