# Active Learning a Convex Body in Low Dimensions

Sariel Har-Peled[1] Mitchell Jones[1] Saladi Rahul[2]

ICALP 2020, July 8–11

[1]University of Illinois at Urbana-Champaign, Urbana, USA

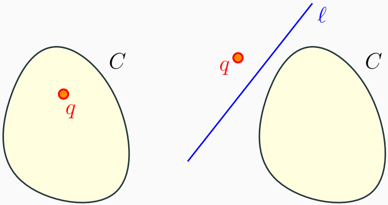
[2]Indian Institute of Science, Bangalore, India

### 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.

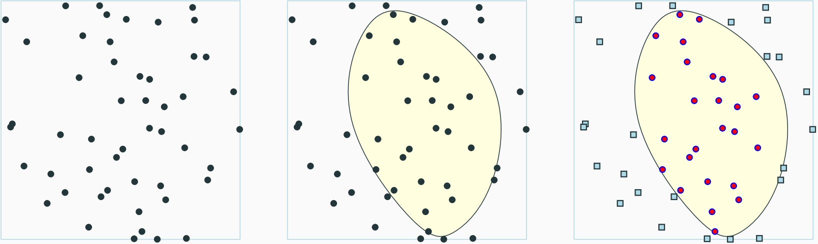


## Problem

**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: 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?

# Additional motivation

- Separation oracles are well-known (OR)

# Additional motivation

- Separation oracles are well-known (OR)
- Computational problems with oracle access:
    - Nearest-neighbor oracles [Har-Peled et al., 2016]
    - Proximity probe [Panahi et al., 2013]
    - Linear queries [Ezra and Sharir, 2019]

# 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

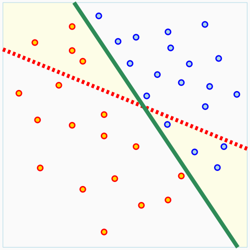
- Allow error in classification
- Algorithm:
    1. Randomly sample input
    2. Obtain labels for sample
    3. Classify sample

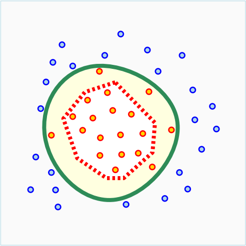
# One approach: PAC learning

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

# One approach: PAC learning

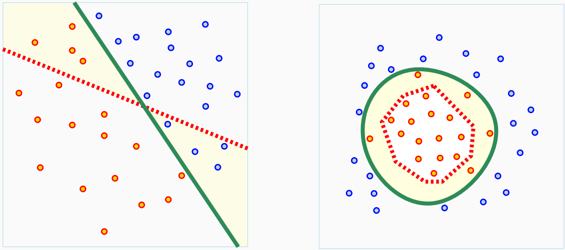
- Misclassified points = symmetric difference of learned and true classifier





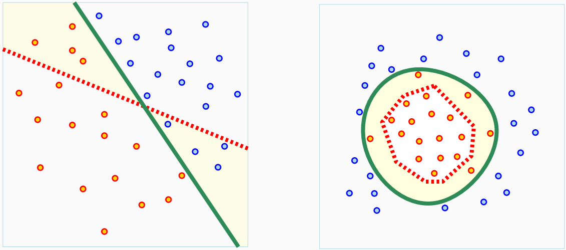
# One approach: PAC learning

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



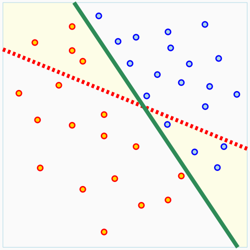
# One approach: PAC learning

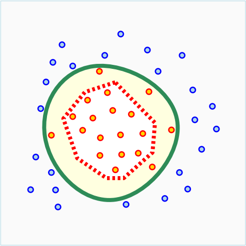
- Misclassified points = symmetric difference of learned and true classifier
- Halfplane $\implies$ symmetric difference is a wedge
- Wedge has finite VC dimension $\implies$ random sample of size $\approx O(\varepsilon^{-1} \log \varepsilon^{-1}) \implies$ $\varepsilon n$ error



# One approach: PAC learning

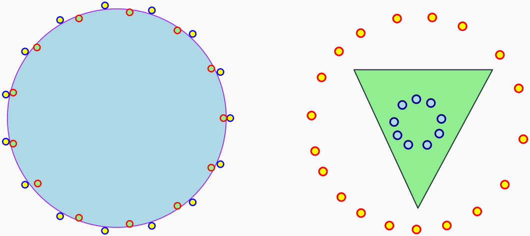
- Misclassified points = symmetric difference of learned and true classifier
- Halfplane $\implies$ symmetric difference is a wedge
- Wedge has finite VC dimension $\implies$ random sample of size $\approx O(\varepsilon^{-1} \log \varepsilon^{-1}) \implies \varepsilon n$ error
- Scheme fails for arbitrary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





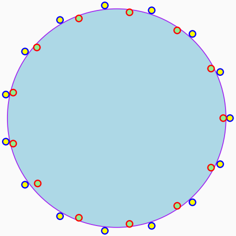
# Hard vs. easy instances

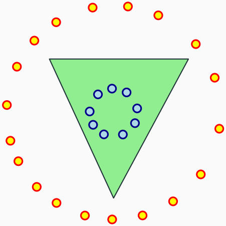
- Worst case: query all points



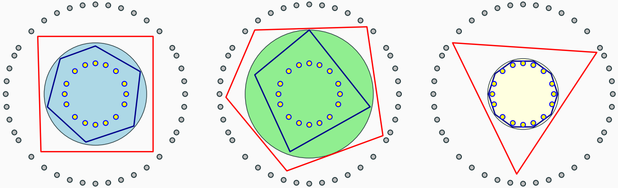
# Hard vs. easy instances

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



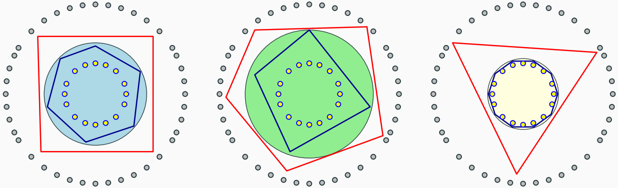


# A lower bound



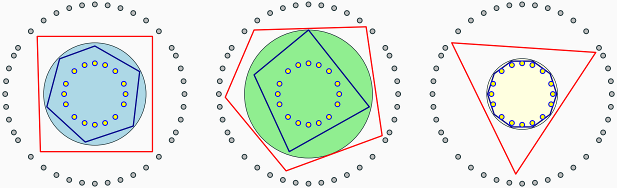
- $F_{\text{in}}$ = convex polygon with fewest vertices s.t. $F_{\text{in}} \subseteq C$ and $C \cap P = F_{\text{in}} \cap P$.

# A lower bound



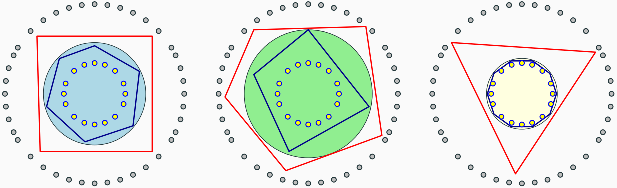
- $F_{\text{in}}$ = convex polygon with fewest vertices s.t. $F_{\text{in}} \subseteq C$ and $C \cap P = F_{\text{in}} \cap P$.
- $F_{\text{out}}$ = convex polygon with fewest vertices s.t. $C \subseteq F_{\text{out}}$ and $C \cap P = F_{\text{out}} \cap P$.

# A lower bound



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



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

# 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) | $\sigma(P, C)$ | $O(k(P) \log n)$ (†) |
| Classify (2D) | $\sigma(P, C)$ | $O(\sigma(P, C) \log^2 n)$ |
| Classify (3D) | — | $O(k(P) \log n)$ (†) |
| Verify in (2D) | $\lvert F_{\mathrm{in}} \rvert$ | $O(\lvert F_{\mathrm{in}} \rvert \log n)$ |
| | | |

(†) $k(P)$ = 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) | $\lvert F_{\text{in}} \rvert$ | $O(\lvert F_{\text{in}} \rvert \log n)$ |
| Verify out (2D) | $\lvert F_{\text{out}} \rvert$ | $O(\lvert F_{\text{out}} \rvert \log n)$ (‡) |

(†) $k(P) =$ 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

# Remarks

**Our result**

The greedy algorithm uses $O(k \log n)$ queries.

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

# Remarks

### Our result

The greedy algorithm uses $O(k \log n)$ queries.

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

- Previously known: $O(k \log k \log n)$ [Kane et al., 2017, inference dimension]

# Remarks

## Our result

The greedy algorithm uses $O(k \log n)$ queries.

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

- Previously known: $O(k \log k \log n)$ [Kane et al., 2017, inference dimension]
- Implementation time:
  $O(n \log^2 n \log \log n + T \cdot k \log n)$, $T =$ query time

# Remarks

### Our result

The greedy algorithm uses $O(k \log n)$ queries.

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

- Previously known: $O(k \log k \log n)$ [Kane et al., 2017, inference dimension]
- Implementation time:
  $O(n \log^2 n \log \log n + T \cdot k \log n)$, $T =$ query time
- $P$ chosen UAR from $[0, 1]^2$
  $\implies \mathbb{E}[k] = \Theta(n^{1/3}) \implies O(n^{1/3} \log n)$

# The greedy algorithm: preliminaries

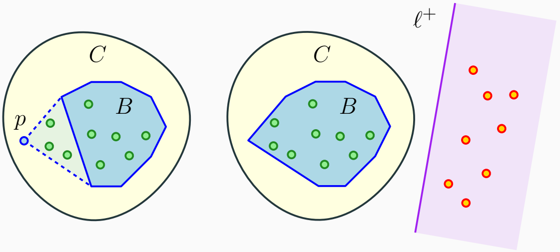
- Maintain approximation $B \subseteq C$

# The greedy algorithm: preliminaries

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

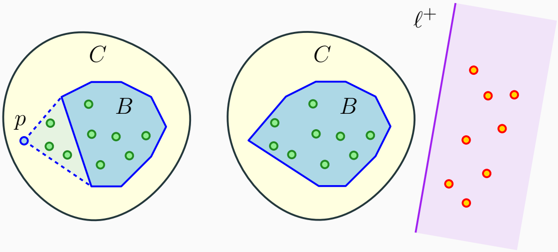
# The greedy algorithm: preliminaries

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



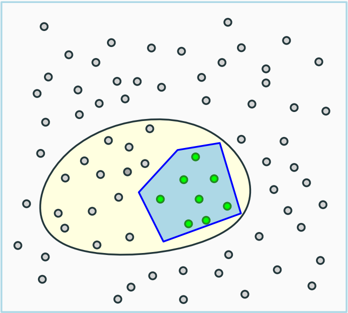
# The greedy algorithm: preliminaries

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



# The greedy algorithm

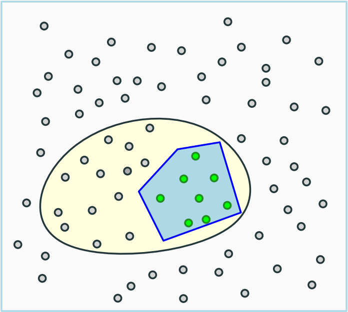
$U \subseteq P$ unclassified points. While $U \neq \varnothing$:



# The greedy algorithm

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

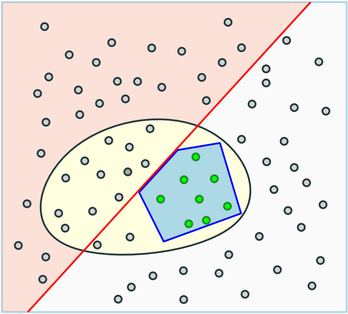
1. $\ell^+ =$ halfspace tangent to $B$ maximizing $|\ell^+ \cap U|$



# The greedy algorithm

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

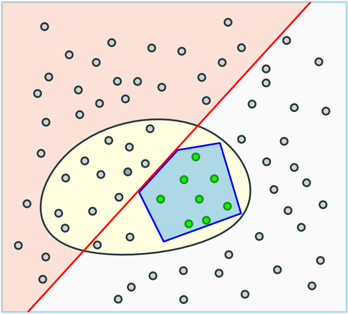
1. $\ell^+$ = halfspace tangent to $B$ maximizing $|\ell^+ \cap U|$



# The greedy algorithm

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

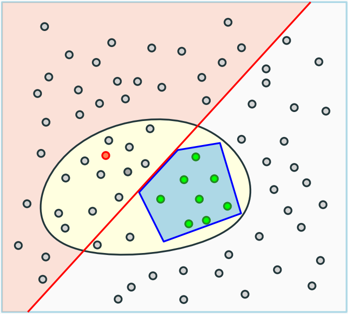
1. $\ell^+ =$ halfspace tangent to $B$ maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$



# The greedy algorithm

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

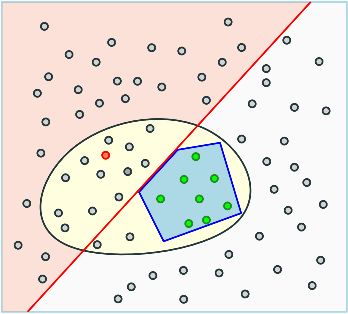
1. $\ell^+$ = halfspace tangent to $B$ maximizing $|\ell^+ \cap U|$
2. $c$ = centerpoint of $\ell^+ \cap U$



# The greedy algorithm

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

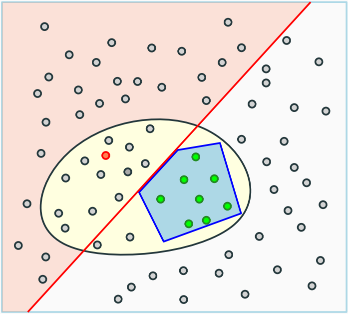
1. $\ell^+$ = halfspace tangent to $B$ maximizing $|\ell^+ \cap U|$
2. $c$ = centerpoint of $\ell^+ \cap U$
3. Query oracle using $c$:



# The greedy algorithm

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

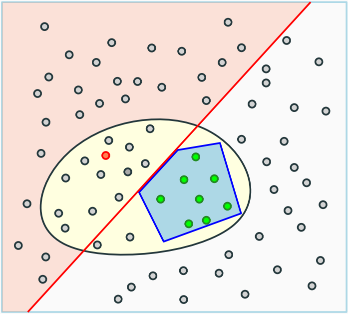
1. $\ell^+ =$ halfspace tangent to $B$ maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$
3. Query oracle using $c$:
   (A) $c \in C \implies$ **expand**$(c)$



# The greedy algorithm

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

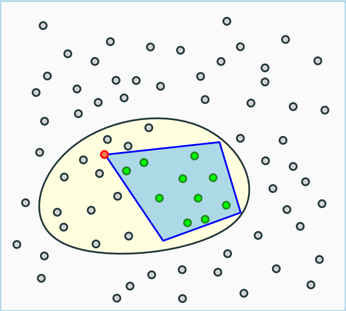
1. $\ell^+$ = halfspace tangent to $B$ maximizing $|\ell^+ \cap U|$
2. $c$ = centerpoint of $\ell^+ \cap U$
3. Query oracle using $c$:
   (A) $c \in C \implies$ **expand**$(c)$
   (B) $c \notin C$, $h$ is a separating line $\implies$ **remove**$(h)$



# The greedy algorithm

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

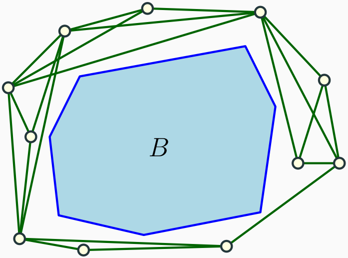
1. $\ell^+ =$ halfspace tangent to $B$ maximizing $|\ell^+ \cap U|$
2. $c =$ centerpoint of $\ell^+ \cap U$
3. Query oracle using $c$:
   (A) $c \in C \implies$ **expand**$(c)$
   (B) $c \notin C$, $h$ is a separating line $\implies$ **remove**$(h)$



# Animation

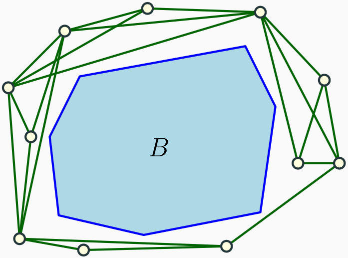
# Analysis

- Count visible pairs of points



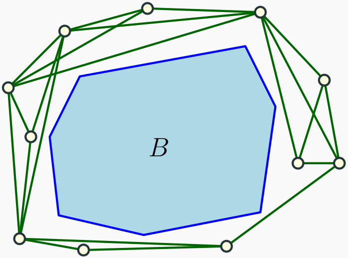
# Analysis

- Count visible pairs of points
- In each iteration:



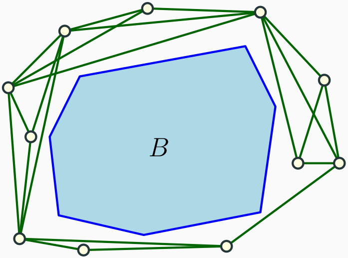
# Analysis

- 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



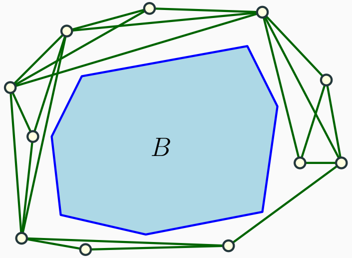
# Analysis

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

**Our result**

The greedy algorithm uses $O(k \log n)$ queries.

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



# Extending the algorithm to 3D

$U \subseteq P$ 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 to 3D

$U \subseteq P$ 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 to 3D

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

# Extending the analysis to 3D

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

# Extending the analysis to 3D

- 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 $B$

# Extending the analysis to 3D

- 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 $B$

## Our result

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

# Conclusions

# Conclusion & open problems

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

# Conclusion & open problems

| Problem | Lowerbound | Upperbound |
| --- | --- | --- |
| Classify (2D) | $\sigma(P, C)$ | $O(k(P) \log n)$<br>$O(\sigma(P, C) \log^2 n)$ |
| 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?

# Conclusion & open problems

| Problem | Lowerbound | Upperbound |
|---|---|---|
| Classify (2D) | $\sigma(P, C)$ | $O(k(P)\log n)$<br>$O(\sigma(P, C)\log^2 n)$ |
| 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) | $\sigma(P, C)$ | $O(k(P) \log n)$<br>$O(\sigma(P, C) \log^2 n)$ |
| 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?
- Higher dimensions?

# Conclusion & open problems

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

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

**Thank you!**

# 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.

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.

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

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