Approximate Polytope Membership Queries and Applications

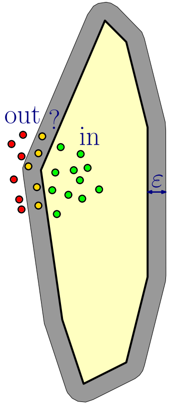
Guilherme D. da Fonseca

Université Clermont Auvergne
LIMOS
INRIA, Sophia-Antipolis
Université de Nice Sophia Antipolis

HDR Defense – June 8, 2018

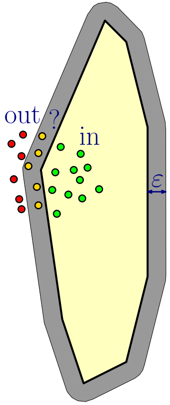
Why study approximate polytope membership?

- Fundamental problem
- Exact solutions are inefficient
- Gives the best known bounds for:
  - Approximate nearest neighbor searching
  - ε-kernel construction
  - Diameter approximation
  - Approximate bichromatic closest pair
  - Minimum Euclidean bottleneck tree approximation
  - ...



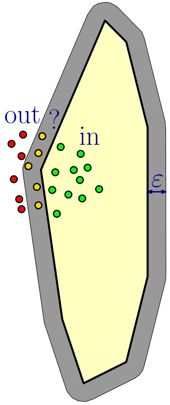
Why study approximate polytope membership?

- Fundamental problem
- Exact solutions are inefficient
- Gives the best known bounds for:
  - Approximate nearest neighbor searching
  - ε-kernel construction
  - Diameter approximation
  - Approximate bichromatic closest pair
  - Minimum Euclidean bottleneck tree approximation
  - ...



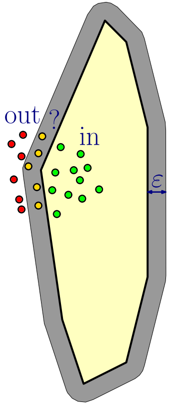
Why study approximate polytope membership?

- Fundamental problem
- Exact solutions are inefficient
- Gives the best known bounds for:
  - Approximate nearest neighbor searching
  - ε-kernel construction
  - Diameter approximation
  - Approximate bichromatic closest pair
  - Minimum Euclidean bottleneck tree approximation
  - ...



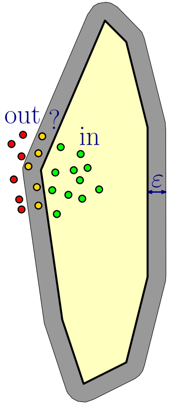
Why study approximate polytope membership?

- Fundamental problem
- Exact solutions are inefficient
- Gives the best known bounds for:
  - Approximate nearest neighbor searching
  - ε-kernel construction
  - Diameter approximation
  - Approximate bichromatic closest pair
  - Minimum Euclidean bottleneck tree approximation
  - ...



Why study approximate polytope membership?

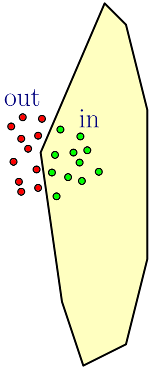
- Fundamental problem
- Exact solutions are inefficient
- Gives the best known bounds for:
  - Approximate nearest neighbor searching
  - ε-kernel construction
  - Diameter approximation
  - Approximate bichromatic closest pair
  - Minimum Euclidean bottleneck tree approximation
  - ...



Exact Polytope Membership Queries

Given a polytope P in d-dimensional space, preprocess P to answer membership queries:

Given a point q, is $q \in P$?

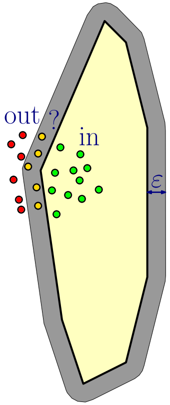


- Assume that dimension d is a constant and P is given as intersection of n halfspaces
- Dual of halfspace emptiness searching
- For $d \leq 3$
  Query time: $O(\log n)$ Storage: $O(n)$
- For $d \geq 4$
  Query time: $O(\log n)$ Storage: $O(n^{\lfloor d/2 \rfloor})$

Approximate Polytope Membership Queries

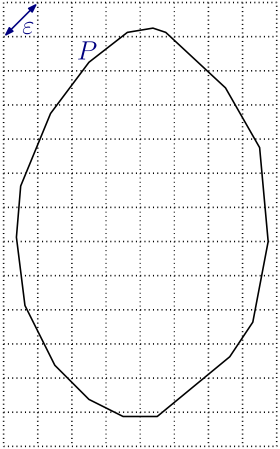
Approximate Version

- An approximation parameter $\varepsilon > 0$ is given
- Assume the polytope has diameter 1
- If the query point's distance from P:
  - 0: answer must be inside
  - $\geq \varepsilon$: answer must be outside
  - > 0 and $< \varepsilon$: either answer is acceptable



- Time-efficient
  Optimal query time: $O(\log \frac{1}{\varepsilon})$
- Space-efficient
  Optimal storage: $O(1/\varepsilon^{(d-1)/2})$

Time Efficient Solution [BFP82]

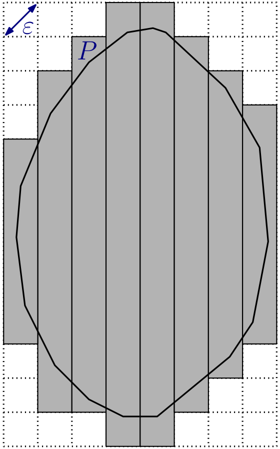


1. Create a grid with cells of size ε
2. For each column, store the topmost and bottommost cells intersecting P
3. Query processing:
   - Locate the column that contains q
   - Compare q with the two extreme values

Time Efficient Solution [BFP82]

- $O(1/\varepsilon^{d-1})$ columns
- Query time: $O(\log \frac{1}{\varepsilon})$ ← optimal
- Storage: $O(1/\varepsilon^{d-1})$ ← not optimal

Time Efficient Solution [BFP82]

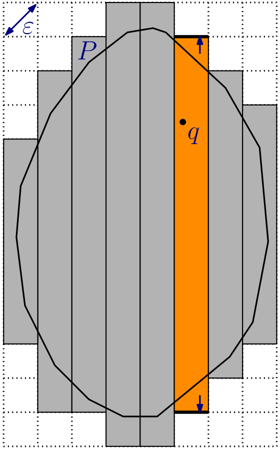


1. Create a grid with cells of size ε
2. For each column, store the topmost and bottommost cells intersecting P
3. Query processing:
   - Locate the column that contains q
   - Compare q with the two extreme values

Time Efficient Solution [BFP82]

- $O(1/\varepsilon^{d-1})$ columns
- Query time: $O(\log \frac{1}{\varepsilon})$ ← optimal
- Storage: $O(1/\varepsilon^{d-1})$ ← not optimal

Time Efficient Solution [BFP82]



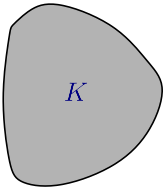
1. Create a grid with cells of size ε
2. For each column, store the topmost and bottommost cells intersecting P
3. Query processing:
   - Locate the column that contains q
   - Compare q with the two extreme values

Time Efficient Solution [BFP82]

- $O(1/\varepsilon^{d-1})$ columns
- Query time: $O(\log \frac{1}{\varepsilon})$ $\leftarrow$ optimal
- Storage: $O(1/\varepsilon^{d-1})$ $\leftarrow$ not optimal

Space Efficient Solution [Dud74]

1. Ball B of radius 2
2. $\sqrt{\varepsilon}$-net N on B
3. Closest point on K for each point in N
4. P bounded by tangent hyperplanes
5. Query processing:
   - Inspect all $O(1/\varepsilon^{\frac{d-1}{2}})$ hyperplanes

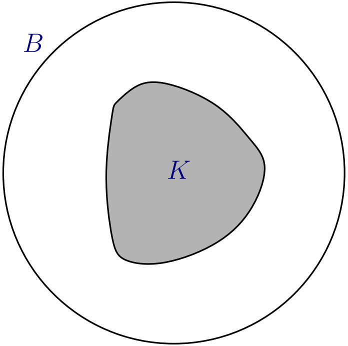


Space Efficient Solution [Dud74]

- Query time: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← not optimal
- Storage: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← optimal

Space Efficient Solution [Dud74]

1. Ball B of radius 2
2. $\sqrt{\varepsilon}$-net N on B
3. Closest point on K for each point in N
4. P bounded by tangent hyperplanes
5. Query processing:
   - Inspect all $O(1/\varepsilon^{\frac{d-1}{2}})$ hyperplanes

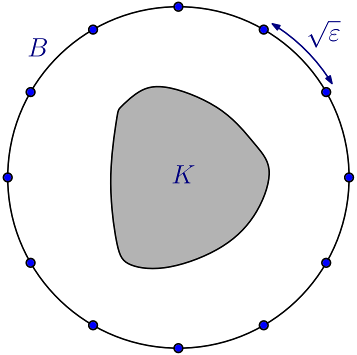


Space Efficient Solution [Dud74]

- Query time: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← not optimal
- Storage: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← optimal

Space Efficient Solution [Dud74]

1. Ball B of radius 2
2. $\sqrt{\varepsilon}$-net N on B
3. Closest point on K for each point in N
4. P bounded by tangent hyperplanes
5. Query processing:
   - Inspect all $O(1/\varepsilon^{\frac{d-1}{2}})$ hyperplanes

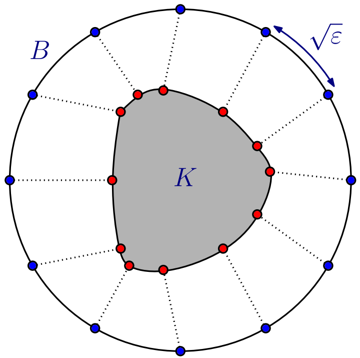


Space Efficient Solution [Dud74]

- Query time: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← not optimal
- Storage: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← optimal

Space Efficient Solution [Dud74]

1. Ball B of radius 2
2. $\sqrt{\varepsilon}$-net N on B
3. Closest point on K for each point in N
4. P bounded by tangent hyperplanes
5. Query processing:
   - Inspect all $O(1/\varepsilon^{\frac{d-1}{2}})$ hyperplanes

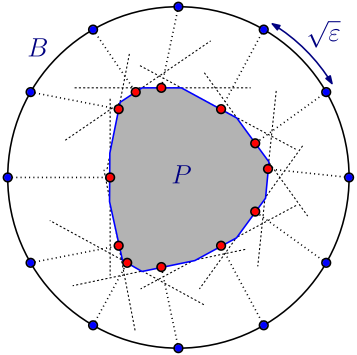


Space Efficient Solution [Dud74]

- Query time: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← not optimal
- Storage: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← optimal

Space Efficient Solution [Dud74]

1. Ball B of radius 2
2. $\sqrt{\varepsilon}$-net N on B
3. Closest point on K for each point in N
4. P bounded by tangent hyperplanes
5. Query processing:
   - Inspect all $O(1/\varepsilon^{\frac{d-1}{2}})$ hyperplanes

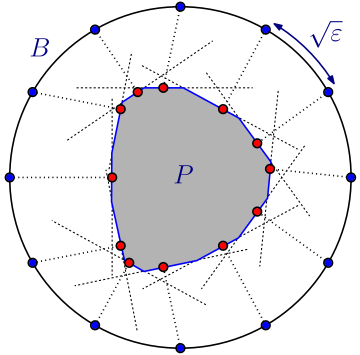


Space Efficient Solution [Dud74]

- Query time: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← not optimal
- Storage: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← optimal

Space Efficient Solution [Dud74]

1. Ball B of radius 2
2. $\sqrt{\varepsilon}$-net N on B
3. Closest point on K for each point in N
4. P bounded by tangent hyperplanes
5. Query processing:
   - Inspect all $O(1/\varepsilon^{\frac{d-1}{2}})$ hyperplanes

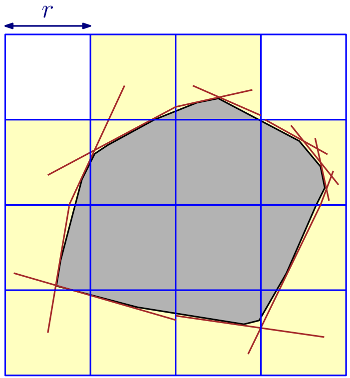


Space Efficient Solution [Dud74]

- Query time: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← not optimal
- Storage: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← optimal

A Simple Tradeoff

1. Generate a grid of size $r \in [\varepsilon, 1]$
2. Preprocessing: For each cell Q intersecting P's boundary:
   - Apply Dudley to $P \cap Q$
   - $O((r/\varepsilon)^{(d-1)/2})$ halfspaces per cell
3. Query Processing:
   - Find the cell containing q
   - Check whether q lies within every halfspace for this cell

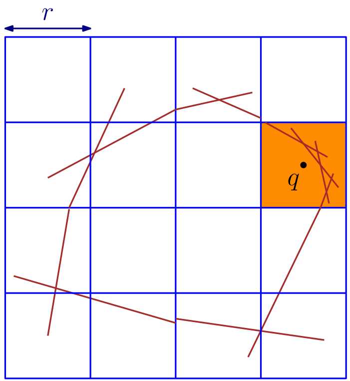


Simple Tradeoff

- Query time: $O((r/\varepsilon)^{(d-1)/2})$
- Storage: $O(1/(r\varepsilon)^{(d-1)/2})$

A Simple Tradeoff

1. Generate a grid of size $r \in [\varepsilon, 1]$
2. Preprocessing: For each cell Q intersecting P's boundary:
   - Apply Dudley to $P \cap Q$
   - $O((r/\varepsilon)^{(d-1)/2})$ halfspaces per cell
3. Query Processing:
   - Find the cell containing q
   - Check whether q lies within every halfspace for this cell

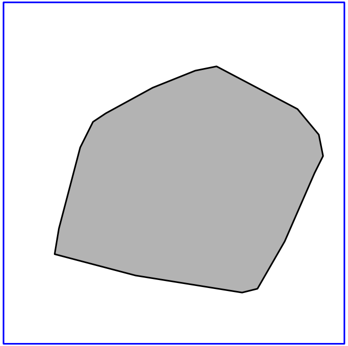


Simple Tradeoff

- Query time: $O((r/\varepsilon)^{(d-1)/2})$
- Storage: $O(1/(r\varepsilon)^{(d-1)/2})$

Split-Reduce Data Structure [AFM18]

$t = 2$



- Input: P, ε, t
- $Q \leftarrow$ unit hypercube
- Split-Reduce(Q)

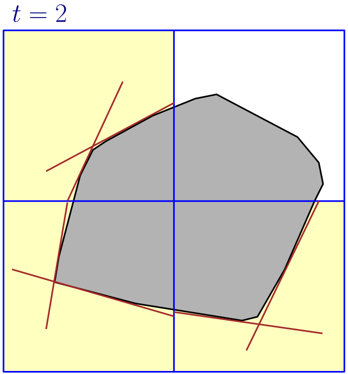
Split-Reduce(Q)

- Find an ε-approximation of $Q \cap P$
- If at most t facets, then Q stores them
- Otherwise, subdivide Q and recurse

Tradeoff

- Query time: $O(t)$
- Storage: ???

Split-Reduce Data Structure [AFM18]



Tradeoff

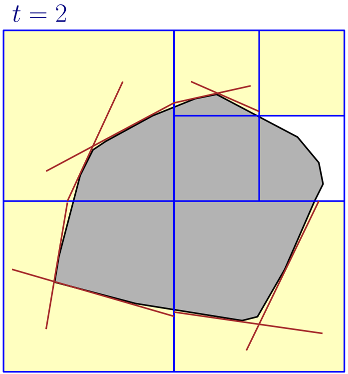
- Query time: $O(t)$
- Storage: ???

- Input: P, ε, t
- $Q \leftarrow$ unit hypercube
- Split-Reduce(Q)

Split-Reduce(Q)

- Find an ε-approximation of $Q \cap P$
- If at most t facets, then Q stores them
- Otherwise, subdivide Q and recurse

Split-Reduce Data Structure [AFM18]



- Input: P, ε, t
- $Q \leftarrow$ unit hypercube
- Split-Reduce(Q)

Split-Reduce(Q)

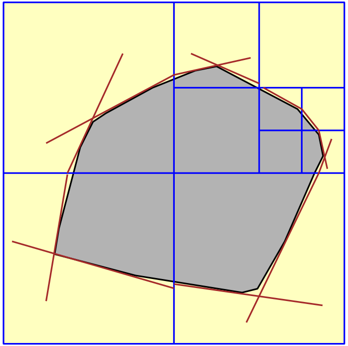
- Find an ε-approximation of $Q \cap P$
- If at most t facets, then Q stores them
- Otherwise, subdivide Q and recurse

Tradeoff

- Query time: $O(t)$
- Storage: ???

Split-Reduce Data Structure [AFM18]

$t = 2$



Tradeoff

- Query time: $O(t)$
- Storage: $???$

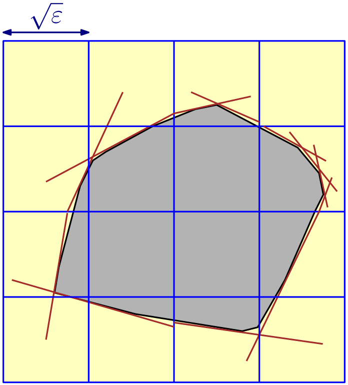
- Input: P, ε, t
- $Q \leftarrow$ unit hypercube
- Split-Reduce(Q)

Split-Reduce(Q)

- Find an ε-approximation of $Q \cap P$
- If at most t facets, then Q stores them
- Otherwise, subdivide Q and recurse

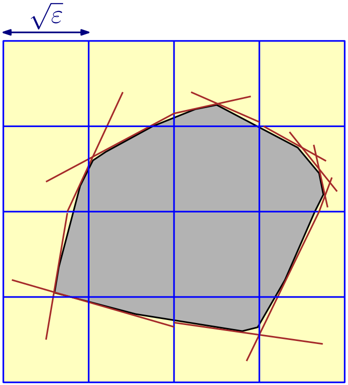
Analysis of Split-Reduce (easy case)

- Easy analysis: $t = 1/\varepsilon^{(d-1)/4}$
- By Dudley in the cell, if diameter $\leq \sqrt{\varepsilon}$, then $O(1/\varepsilon^{(d-1)/4})$ halfspaces suffice
- Cells of size $\sqrt{\varepsilon}$ are not subdivided
- Each Dudley halfspace is only useful within a radius of $\sqrt{\varepsilon}$
- It hits $O(1)$ cells of size $\sqrt{\varepsilon}$
- Total number of halfspaces: $O(1/\varepsilon^{(d-1)/2})$



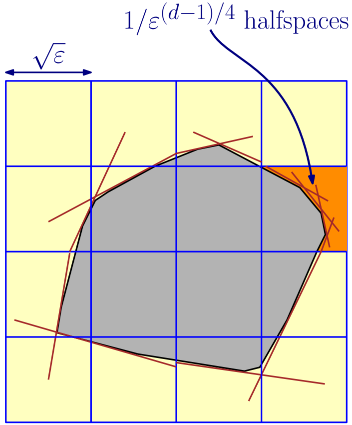
Analysis of Split-Reduce (easy case)

- Easy analysis: $t = 1/\varepsilon^{(d-1)/4}$
- By Dudley in the cell, if diameter $\leq \sqrt{\varepsilon}$, then $O(1/\varepsilon^{(d-1)/4})$ halfspaces suffice
- Cells of size $\sqrt{\varepsilon}$ are not subdivided
- Each Dudley halfspace is only useful within a radius of $\sqrt{\varepsilon}$
- It hits $O(1)$ cells of size $\sqrt{\varepsilon}$
- Total number of halfspaces: $O(1/\varepsilon^{(d-1)/2})$

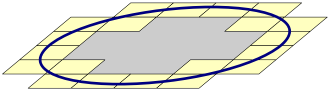


Analysis of Split-Reduce (easy case)

- Easy analysis: $t = 1/\varepsilon^{(d-1)/4}$
- By Dudley in the cell, if diameter $\leq \sqrt{\varepsilon}$, then $O(1/\varepsilon^{(d-1)/4})$ halfspaces suffice
- Cells of size $\sqrt{\varepsilon}$ are not subdivided
- Each Dudley halfspace is only useful within a radius of $\sqrt{\varepsilon}$
- It hits $O(1)$ cells of size $\sqrt{\varepsilon}$
- Total number of halfspaces: $O(1/\varepsilon^{(d-1)/2})$



Lower bound to Split-Reduce

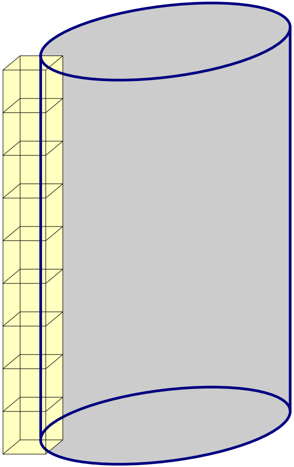


- Place a small enough ball in $\mathbb{R}^k$
- High curvature forces small cells
- No problem: small diameter
- Extrude the ball in $d - k$ dimensions
- Quadtree cells are hypercubes
- Too many cells!

- What if cells are not hypercubes?

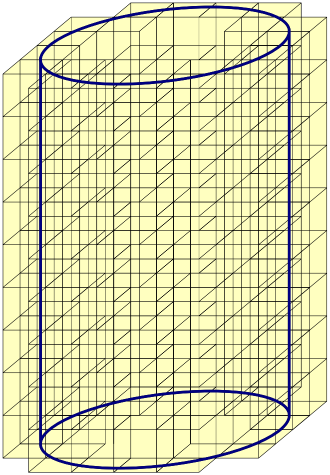
Lower bound to Split-Reduce

- Place a small enough ball in $\mathbb{R}^k$
- High curvature forces small cells
- No problem: small diameter
- Extrude the ball in $d - k$ dimensions
- Quadtree cells are hypercubes
- Too many cells!
- What if cells are not hypercubes?



Lower bound to Split-Reduce

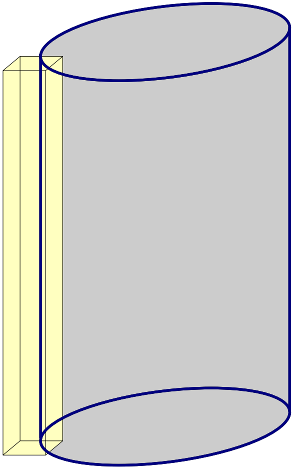
- Place a small enough ball in $\mathbb{R}^k$
- High curvature forces small cells
- No problem: small diameter
- Extrude the ball in $d-k$ dimensions
- Quadtree cells are hypercubes
- Too many cells!
- What if cells are not hypercubes?



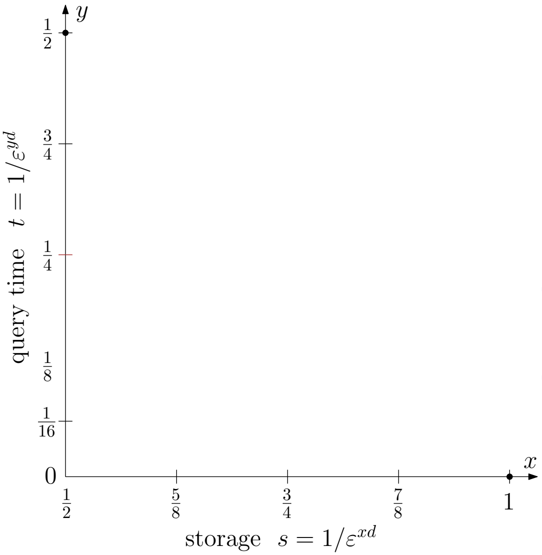
Lower bound to Split-Reduce

- Place a small enough ball in $\mathbb{R}^k$
- High curvature forces small cells
- No problem: small diameter
- Extrude the ball in $d-k$ dimensions
- Quadtree cells are hypercubes
- Too many cells!

- What if cells are not hypercubes?



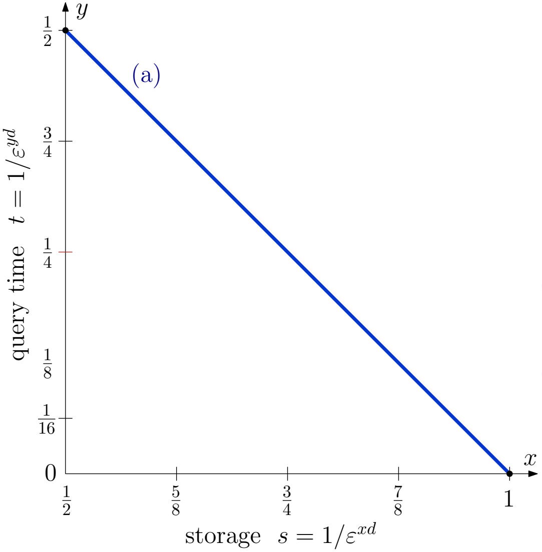
General Tradeoff



- Tight analysis is an open problem
- Best analysis is very complex

(a) Simple tradeoff
(b) Easy $t = 1/\varepsilon^{(d-1)/4}$ case
(c) Best upper bound
(d) Lower bound to Split-Reduce
(e) Next data structure: uses Macbeath regions!

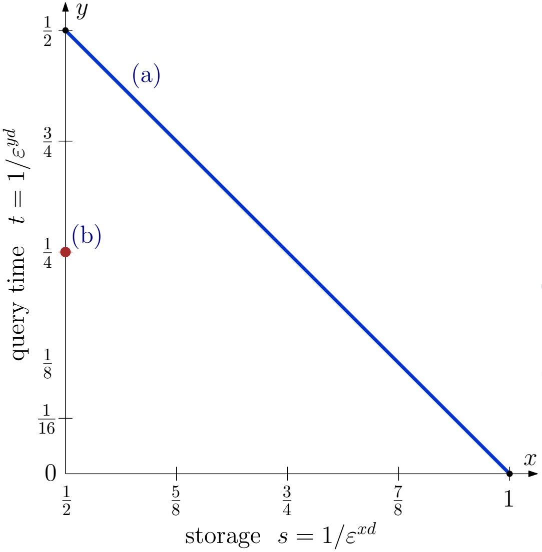
General Tradeoff



- Tight analysis is an open problem
- Best analysis is very complex

(a) Simple tradeoff
(b) Easy $t = 1/\varepsilon^{(d-1)/4}$ case
(c) Best upper bound
(d) Lower bound to Split-Reduce
(e) Next data structure: uses Macbeath regions!

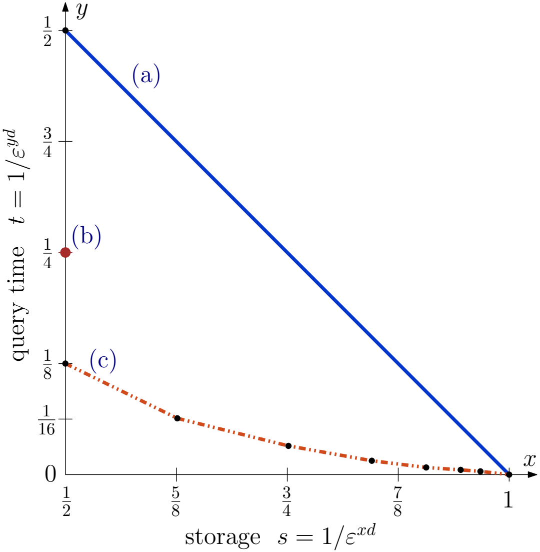
General Tradeoff



- Tight analysis is an open problem
- Best analysis is very complex

(a) Simple tradeoff
(b) Easy $t = 1/\varepsilon^{(d-1)/4}$ case
(c) Best upper bound
(d) Lower bound to Split-Reduce
(e) Next data structure: uses Macbeath regions!

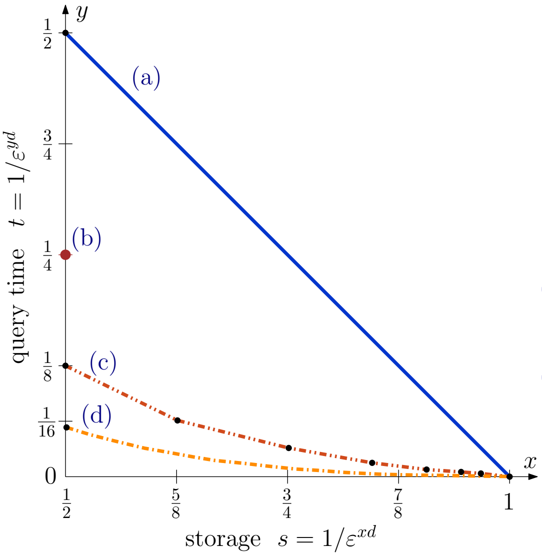
General Tradeoff



- Tight analysis is an open problem
- Best analysis is very complex

(a) Simple tradeoff
(b) Easy $t = 1/\varepsilon^{(d-1)/4}$ case
(c) Best upper bound
(d) Lower bound to Split-Reduce
(e) Next data structure:
uses Macbeath regions!

General Tradeoff



- Tight analysis is an open problem
- Best analysis is very complex

(a) Simple tradeoff

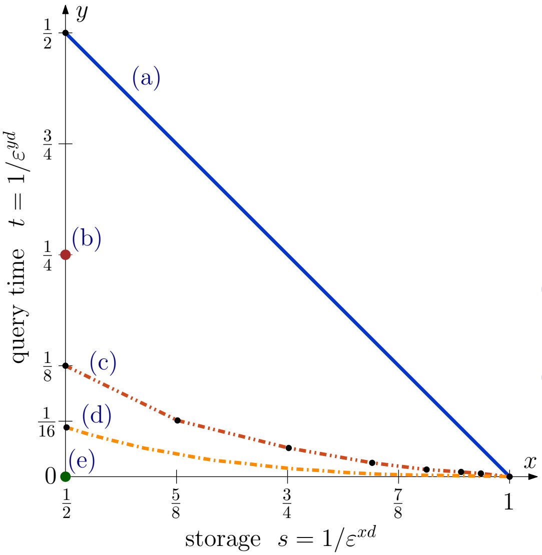
(b) Easy $t = 1/\varepsilon^{(d-1)/4}$ case

(c) Best upper bound

(d) Lower bound to Split-Reduce

(e) Next data structure: uses Macbeath regions!

General Tradeoff



- Tight analysis is an open problem
- Best analysis is very complex

(a) Simple tradeoff

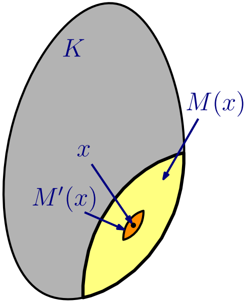
(b) Easy $t = 1/\varepsilon^{(d-1)/4}$ case

(c) Best upper bound

(d) Lower bound to Split-Reduce

(e) Next data structure: uses Macbeath regions!

Macbeath Regions [Mac52]



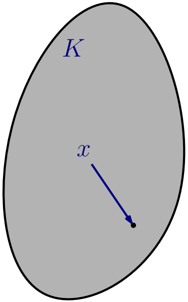
Given a convex body K, $x \in K$, and $\lambda > 0$:

- $M^\lambda(x) = x + \lambda((K - x) \cap (x - K))$
- $M(x) = M^1(x)$: intersection of K and K reflected around x
- $M'(x) = M^{1/5}(x)$

Properties

- $M'(x) \cap M'(y) \neq \emptyset \ \Rightarrow \ M'(x) \subseteq M(y)$
- $y \in M'(x) \ \Rightarrow \ \delta(y) = \Theta(\delta(x))$
- $\delta(x)$: distance from x to ∂K

Macbeath Regions [Mac52]



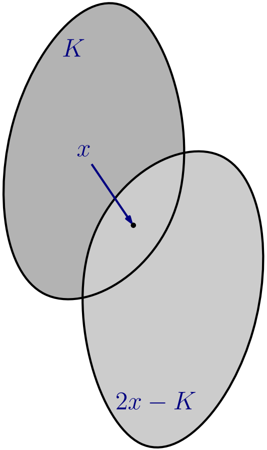
Given a convex body K, $x \in K$, and $\lambda > 0$:

- $M^\lambda(x) = x + \lambda((K - x) \cap (x - K))$
- $M(x) = M^1(x)$: intersection of K and K reflected around x
- $M'(x) = M^{1/5}(x)$

Properties

- $M'(x) \cap M'(y) \neq \emptyset \;\Rightarrow\; M'(x) \subseteq M(y)$
- $y \in M'(x) \;\Rightarrow\; \delta(y) = \Theta(\delta(x))$
- $\delta(x)$: distance from x to ∂K

Macbeath Regions [Mac52]



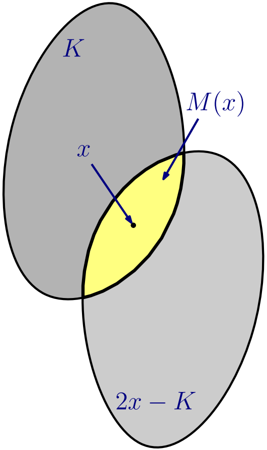
Given a convex body K, $x \in K$, and $\lambda > 0$:

- $M^\lambda(x) = x + \lambda((K - x) \cap (x - K))$
- $M(x) = M^1(x)$: intersection of K and K reflected around x
- $M'(x) = M^{1/5}(x)$

Properties

- $M'(x) \cap M'(y) \neq \emptyset \;\Rightarrow\; M'(x) \subseteq M(y)$
- $y \in M'(x) \;\Rightarrow\; \delta(y) = \Theta(\delta(x))$
- $\delta(x)$: distance from x to ∂K

Macbeath Regions [Mac52]



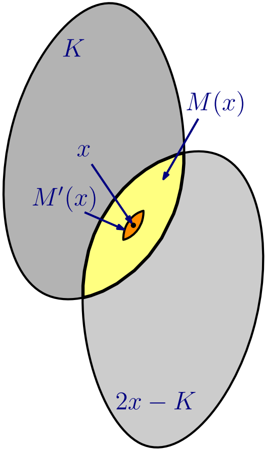
Given a convex body K, $x \in K$, and $\lambda > 0$:

- $M^\lambda(x) = x + \lambda((K - x) \cap (x - K))$
- $M(x) = M^1(x)$: intersection of K and K reflected around x
- $M'(x) = M^{1/5}(x)$

Properties

- $M'(x) \cap M'(y) \neq \emptyset \;\Rightarrow\; M'(x) \subseteq M(y)$
- $y \in M'(x) \;\Rightarrow\; \delta(y) = \Theta(\delta(x))$
- $\delta(x)$: distance from x to ∂K

Macbeath Regions [Mac52]



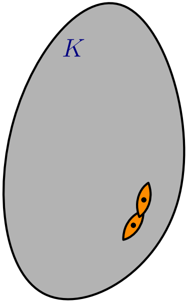
Given a convex body K, $x \in K$, and $\lambda > 0$:

- $M^\lambda(x) = x + \lambda((K - x) \cap (x - K))$
- $M(x) = M^1(x)$: intersection of K and K reflected around x
- $M'(x) = M^{1/5}(x)$

Properties

- $M'(x) \cap M'(y) \neq \emptyset \;\Rightarrow\; M'(x) \subseteq M(y)$
- $y \in M'(x) \;\Rightarrow\; \delta(y) = \Theta(\delta(x))$
- $\delta(x)$: distance from x to ∂K

Macbeath Regions [Mac52]



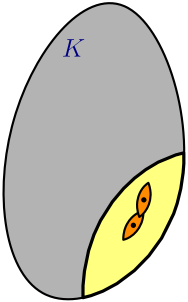
Given a convex body K, $x \in K$, and $\lambda > 0$:

- $M^{\lambda}(x) = x + \lambda((K - x) \cap (x - K))$
- $M(x) = M^1(x)$: intersection of K and K reflected around x
- $M'(x) = M^{1/5}(x)$

Properties

- $M'(x) \cap M'(y) \neq \emptyset \;\Rightarrow\; M'(x) \subseteq M(y)$
- $y \in M'(x) \;\Rightarrow\; \delta(y) = \Theta(\delta(x))$
- $\delta(x)$: distance from x to ∂K

Macbeath Regions [Mac52]



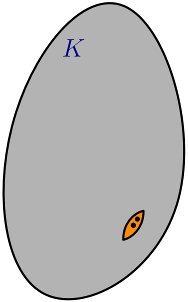
Given a convex body K, $x \in K$, and $\lambda > 0$:

- $M^\lambda(x) = x + \lambda((K - x) \cap (x - K))$
- $M(x) = M^1(x)$: intersection of K and K reflected around x
- $M'(x) = M^{1/5}(x)$

Properties

- $M'(x) \cap M'(y) \neq \emptyset \;\Rightarrow\; M'(x) \subseteq M(y)$
- $y \in M'(x) \;\Rightarrow\; \delta(y) = \Theta(\delta(x))$
- $\delta(x)$: distance from x to ∂K

Macbeath Regions [Mac52]



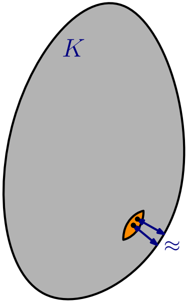
Given a convex body K, $x \in K$, and $\lambda > 0$:

- $M^\lambda(x) = x + \lambda((K - x) \cap (x - K))$
- $M(x) = M^1(x)$: intersection of K and K reflected around x
- $M'(x) = M^{1/5}(x)$

Properties

- $M'(x) \cap M'(y) \neq \emptyset \;\Rightarrow\; M'(x) \subseteq M(y)$
- $y \in M'(x) \;\Rightarrow\; \delta(y) = \Theta(\delta(x))$
- $\delta(x)$: distance from x to ∂K

Macbeath Regions [Mac52]



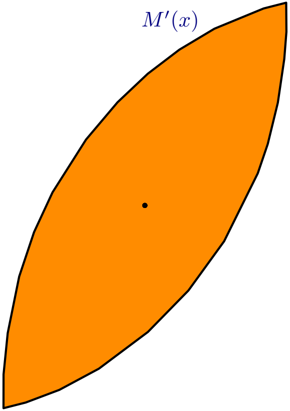
Given a convex body K, $x \in K$, and $\lambda > 0$:

- $M^\lambda(x) = x + \lambda((K - x) \cap (x - K))$
- $M(x) = M^1(x)$: intersection of K and K reflected around x
- $M'(x) = M^{1/5}(x)$

Properties

- $M'(x) \cap M'(y) \neq \emptyset \;\Rightarrow\; M'(x) \subseteq M(y)$
- $y \in M'(x) \;\Rightarrow\; \delta(y) = \Theta(\delta(x))$
- $\delta(x)$: distance from x to ∂K

Macbeath Ellipsoids



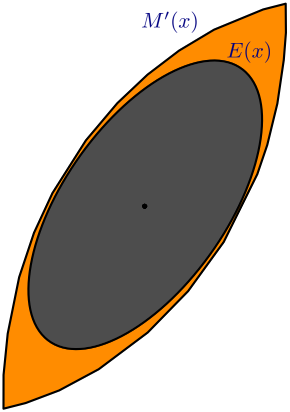
John Ellipsoid [Joh48]

For every centrally symmetric convex body K in $\mathbb{R}^d$, there exist ellipsoids E_1, E_2 such that $E_1 \subseteq K \subseteq E_2$ and E_2 is a $\sqrt{d}$-scaling of E_1

Macbeath Ellipsoid

- $E(x)$: enclosed John ellipsoid of $M'(x)$
- $M^\lambda(x) \subseteq E(x) \subseteq M'(x)$ for $\lambda = 1/(5\sqrt{d})$

Macbeath Ellipsoids



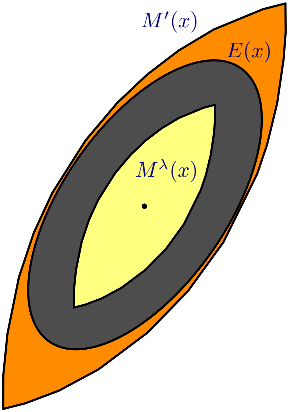
John Ellipsoid [Joh48]

For every centrally symmetric convex body K in $\mathbb{R}^d$, there exist ellipsoids E_1, E_2 such that $E_1 \subseteq K \subseteq E_2$ and E_2 is a $\sqrt{d}$-scaling of E_1

Macbeath Ellipsoid

- $E(x)$: enclosed John ellipsoid of $M'(x)$
- $M^\lambda(x) \subseteq E(x) \subseteq M'(x)$ for $\lambda = 1/(5\sqrt{d})$

Macbeath Ellipsoids



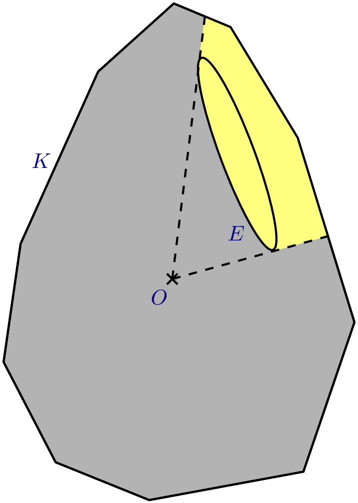
John Ellipsoid [Joh48]

For every centrally symmetric convex body K in $\mathbb{R}^d$, there exist ellipsoids E_1, E_2 such that $E_1 \subseteq K \subseteq E_2$ and E_2 is a $\sqrt{d}$-scaling of E_1

Macbeath Ellipsoid

- $E(x)$: enclosed John ellipsoid of $M'(x)$
- $M^\lambda(x) \subseteq E(x) \subseteq M'(x)$ for $\lambda = 1/(5\sqrt{d})$

Shadow of Macbeath Ellipsoids



Shadow of ellipsoid E

Points $p \in K$ such that ray Op intersects E

- Reaches the boundary
- Directional width: similar to E

Covering with Macbeath Ellipsoids

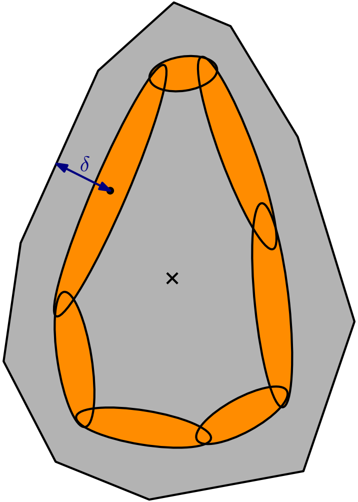
Covering (see [Bar07])

Given:

- K: convex body
- δ: small positive parameter

There exist ellipsoids $E(x_1), \ldots, E(x_k)$

- $\delta(x_1) = \cdots = \delta(x_k) = \delta$
- Cover: Shadows cover the boundary
- $k = O(1/\delta^{(d-1)/2})$ [AFM17c]



Covering with Macbeath Ellipsoids

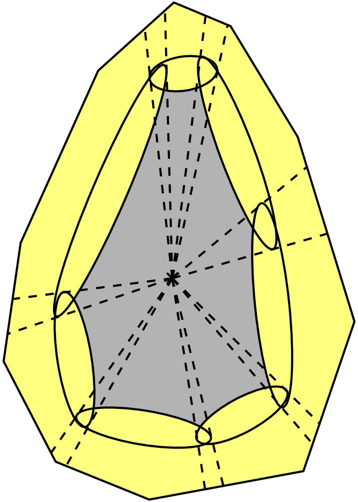
Covering (see [Bar07])

Given:

- K: convex body
- δ: small positive parameter

There exist ellipsoids $E(x_1), \ldots, E(x_k)$

- $\delta(x_1) = \cdots = \delta(x_k) = \delta$
- Cover: Shadows cover the boundary
- $k = O(1/\delta^{(d-1)/2})$ [AFM17c]



Covering with Macbeath Ellipsoids

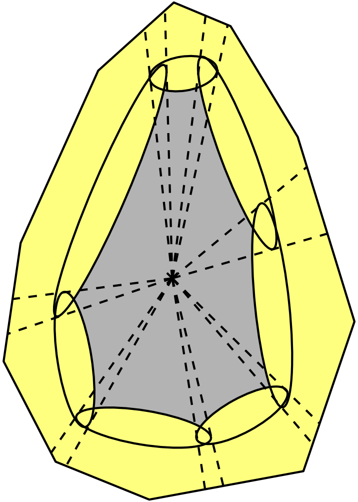
Covering (see [Bar07])

Given:

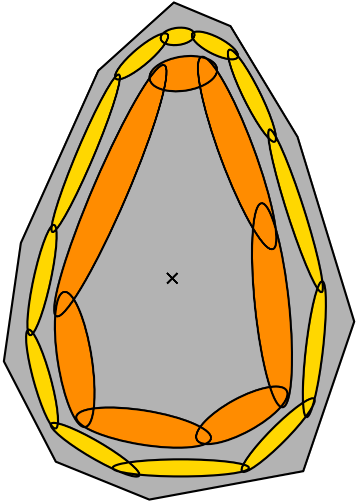
- K: convex body
- δ: small positive parameter

There exist ellipsoids $E(x_1), \ldots, E(x_k)$

- $\delta(x_1) = \cdots = \delta(x_k) = \delta$
- Cover: Shadows cover the boundary
- $k = O(1/\delta^{(d-1)/2})$ [AFM17c]



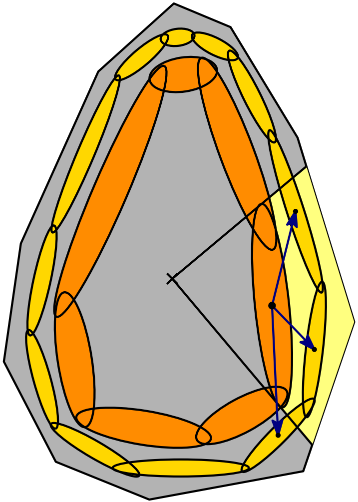
Hierarchy of Macbeath Ellipsoids [AFM17a]



Hierarchy

- Each level i a δ_i-covering
- $\ell = \Theta(\log \frac{1}{\varepsilon})$ levels
- $\delta_0 = \Theta(1)$, $\delta_\ell = \Theta(\varepsilon)$
- $\delta_{i+1} = \delta_i/2$
- E is parent of E' if
  - Levels are consecutive
  - Shadow of E intersects E'
- Each node has $O(1)$ children

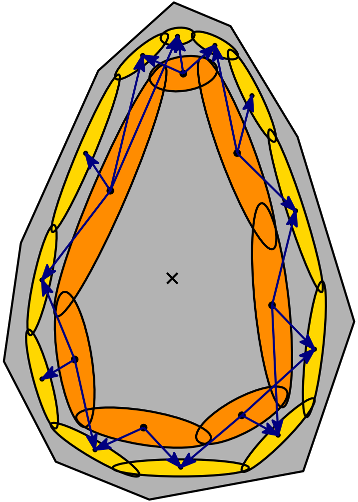
Hierarchy of Macbeath Ellipsoids [AFM17a]



Hierarchy

- Each level i a δ_i-covering
- $\ell = \Theta(\log \frac{1}{\varepsilon})$ levels
- $\delta_0 = \Theta(1)$, $\delta_\ell = \Theta(\varepsilon)$
- $\delta_{i+1} = \delta_i/2$
- E is parent of E' if
  - Levels are consecutive
  - Shadow of E intersects E'
- Each node has $O(1)$ children

Hierarchy of Macbeath Ellipsoids [AFM17a]



Hierarchy

- Each level i a δ_i-covering
- $\ell = \Theta(\log \frac{1}{\varepsilon})$ levels
- $\delta_0 = \Theta(1)$, $\delta_\ell = \Theta(\varepsilon)$
- $\delta_{i+1} = \delta_i / 2$
- E is parent of E' if
  - Levels are consecutive
  - Shadow of E intersects E'
- Each node has $O(1)$ children

Ray Shooting from the Origin

Ray Shooting from the Origin (generalizes polytope membership)

Preprocess:

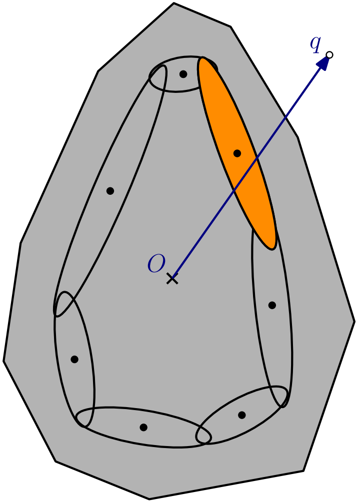
- K: convex body
- ε: small positive parameter

Query:

- Oq: ray from the origin towards q

Query algorithm:

- Find an ellipsoid intersecting Oq at level 0
- Repeat among children at next level
- Stop at leaf node
- Leaf ellipsoid ε-approximates boundary



Ray Shooting from the Origin

Ray Shooting from the Origin (generalizes polytope membership)

Preprocess:

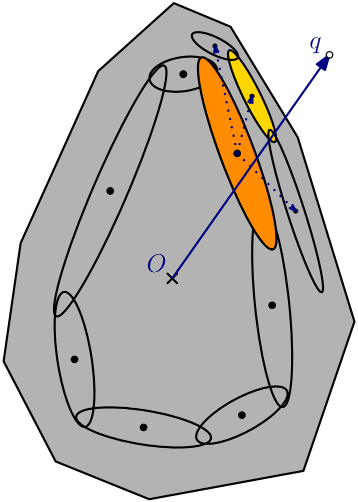
- K: convex body
- ε: small positive parameter

Query:

- Oq: ray from the origin towards q

Query algorithm:

- Find an ellipsoid intersecting Oq at level 0
- Repeat among children at next level
- Stop at leaf node
- Leaf ellipsoid ε-approximates boundary



Ray Shooting from the Origin

Ray Shooting from the Origin (generalizes polytope membership)

Preprocess:

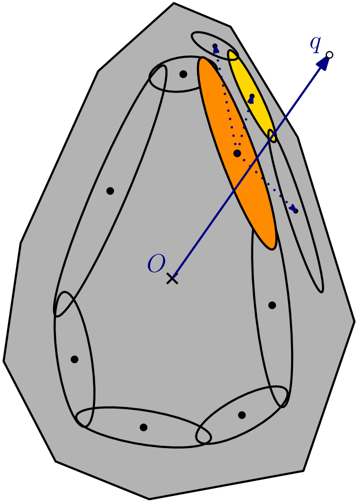
- K: convex body
- ε: small positive parameter

Query:

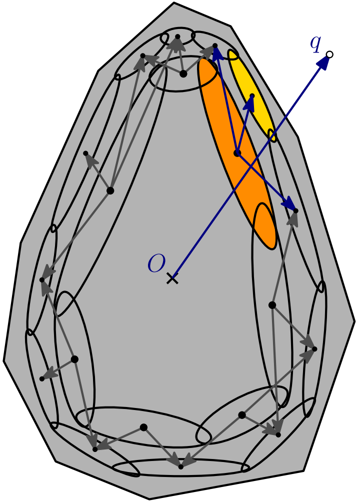
- Oq: ray from the origin towards q

Query algorithm:

- Find an ellipsoid intersecting Oq at level 0
- Repeat among children at next level
- Stop at leaf node
- Leaf ellipsoid ε-approximates boundary



Analysis



- Out-degree: $O(1)$
- Query time per level: $O(1)$
- Number of levels: $O(\log \frac{1}{\varepsilon})$

Query time

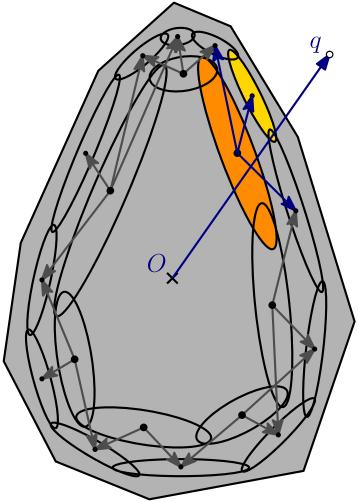
- $O(\log \frac{1}{\varepsilon})$ ← optimal

- Storage for bottom level: $O(1/\varepsilon^{(d-1)/2})$
- Geometric progression of storage per level

Storage

- $O(1/\varepsilon^{(d-1)/2})$ ← optimal

Analysis



- Out-degree: $O(1)$
- Query time per level: $O(1)$
- Number of levels: $O(\log \frac{1}{\varepsilon})$

Query time

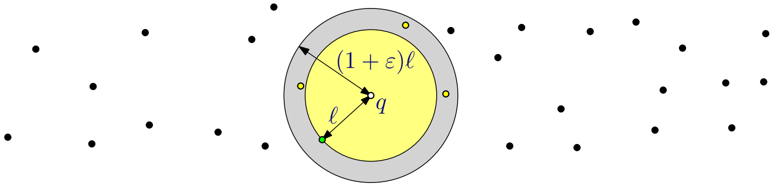
- $O(\log \frac{1}{\varepsilon})$ ← optimal

- Storage for bottom level: $O(1/\varepsilon^{(d-1)/2})$
- Geometric progression of storage per level

Storage

- $O(1/\varepsilon^{(d-1)/2})$ ← optimal

Approximate Nearest (ANN) Neighbor Searching



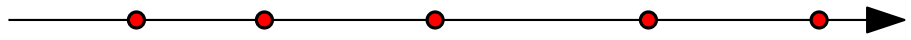
Approximate Nearest Neighbor

Preprocess n points such that, given a query point q, we can find a point within at most $1+\varepsilon$ times the distance to q's nearest neighbor

- Applications to pattern recognition, machine learning, computer vision...
- Huge literature (theory, applications, heuristics...)

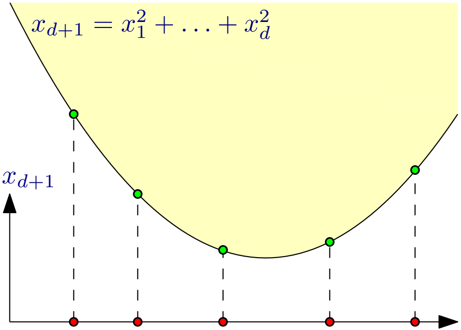
Lifting

- Exact nearest neighbor reduces to ray shooting
- Dimension increases by 1
- Each data point is lifted into a paraboloid
- Polyhedron defined by tangent hyperplanes
- Query: vertical ray shooting



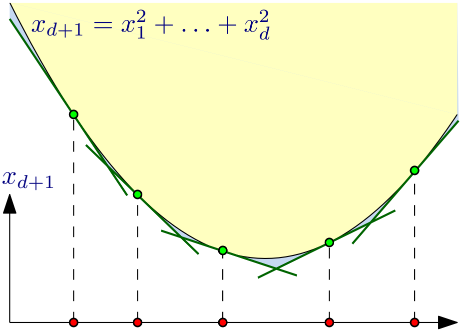
Lifting

- Exact nearest neighbor reduces to ray shooting
- Dimension increases by 1
- Each data point is lifted into a paraboloid
- Polyhedron defined by tangent hyperplanes
- Query: vertical ray shooting



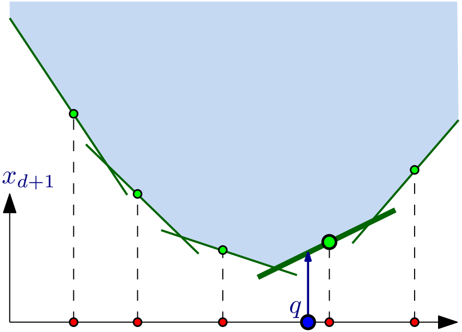
Lifting

- Exact nearest neighbor reduces to ray shooting
- Dimension increases by 1
- Each data point is lifted into a paraboloid
- Polyhedron defined by tangent hyperplanes
- Query: vertical ray shooting



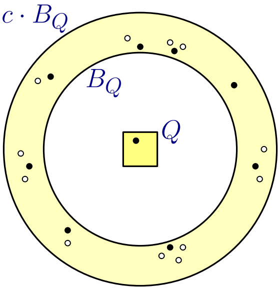
Lifting

- Exact nearest neighbor reduces to ray shooting
- Dimension increases by 1
- Each data point is lifted into a paraboloid
- Polyhedron defined by tangent hyperplanes
- Query: vertical ray shooting



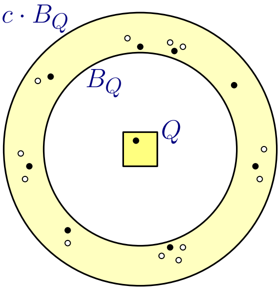
Reduction to Approximate Polytope Membership [AFM18]

- Polyhedron is unbounded
- Unbounded approximation error
- Solution: separation
- Partition space into cells such that: [AMM09]
  - Each cell Q is associated with candidates to be the ANN for query points in Q
  - Total number of candidates is $\widetilde{O}(n)$
  - All but 1 candidate are inside a constant-radius annulus

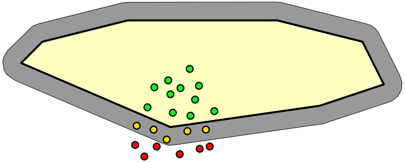


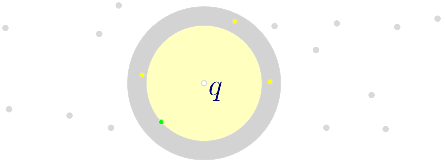
Reduction to Approximate Polytope Membership [AFM18]

- Polyhedron is unbounded
- Unbounded approximation error
- Solution: separation
- Partition space into cells such that: [AMM09]
  - Each cell Q is associated with candidates to be the ANN for query points in Q
  - Total number of candidates is $\widetilde{O}(n)$
  - All but 1 candidate are inside a constant-radius annulus



Reduction





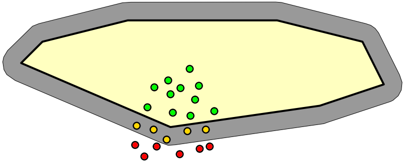
Given APM

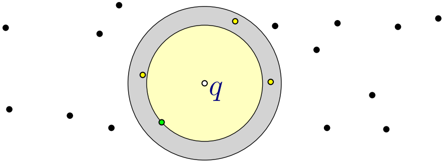
- $d+1$ dimensions
- Query time: at most t
- Storage: s
- Preprocessing: $O(n \log \frac{1}{\varepsilon} + b)$
- t, s, b: functions of ε

Resulting ANN

- d dimensions
- Query time: $O(\log n + t \cdot \log \frac{1}{\varepsilon})$
- Storage: $O(n \log \frac{1}{\varepsilon} + n \cdot s/t)$
- Preprocessing: $O(n \log n \log \frac{1}{\varepsilon} + n \cdot b/t)$

Reduction





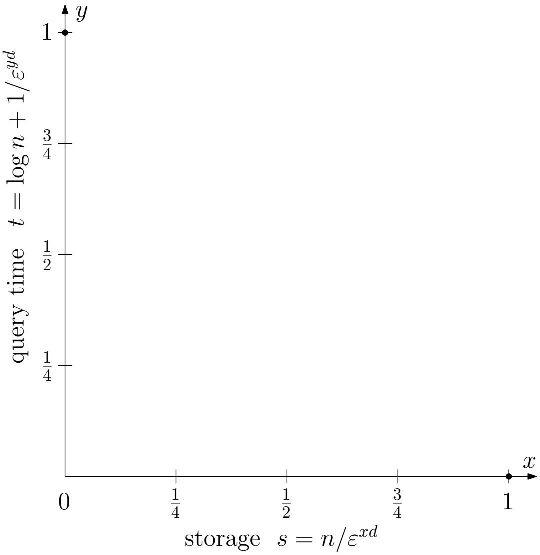
Given APM

- $d+1$ dimensions
- Query time: at most t
- Storage: s
- Preprocessing: $O(n \log \frac{1}{\varepsilon} + b)$
- t, s, b: functions of ε

Resulting ANN

- d dimensions
- Query time: $O(\log n + t \cdot \log \frac{1}{\varepsilon})$
- Storage: $O(n \log \frac{1}{\varepsilon} + n \cdot s/t)$
- Preprocessing: $O(n \log n \log \frac{1}{\varepsilon} + n \cdot b/t)$

Space-Time Tradeoffs for ANN



(a) First generation (before 2002)

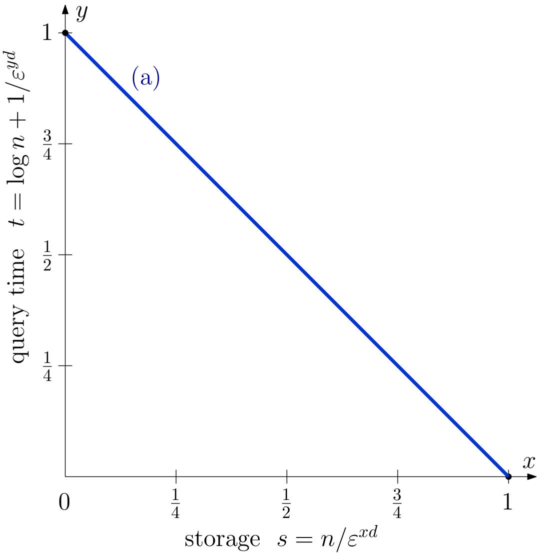
(b) AVDs [AMM09]

(c) Reduction to Split-Reduce

(d) Reduction to Macbeath regions

Best Upper Bound

- For $\log \frac{1}{\varepsilon} \leq m \leq 1/\varepsilon^{d/2}$
  Query time: $O(\log n + 1/(m\,\varepsilon^{d/2}))$
  Storage: $O(n\,m)$
- Setting $m = 1/\varepsilon^{d/2}$
  Query time: $O(\log n)$
  Storage: $O(n/\varepsilon^{d/2})$

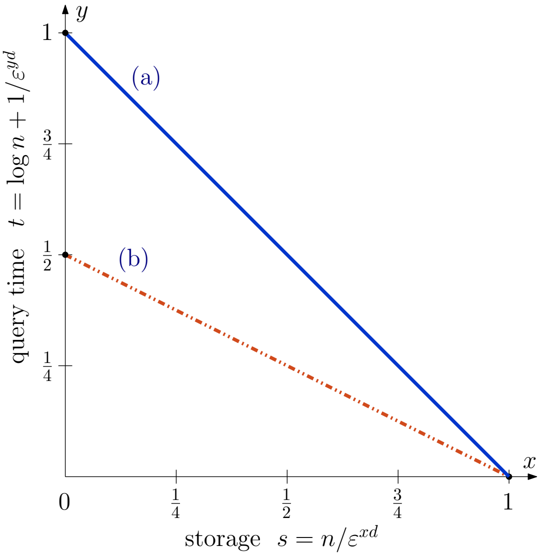


(a) First generation (before 2002)
(b) AVDs [AMM09]
(c) Reduction to Split-Reduce
(d) Reduction to Macbeath regions

Best Upper Bound

- For $\log \frac{1}{\varepsilon} \leq m \leq 1/\varepsilon^{d/2}$
  Query time: $O(\log n + 1/(m\,\varepsilon^{d/2}))$
  Storage: $O(n\,m)$
- Setting $m = 1/\varepsilon^{d/2}$
  Query time: $O(\log n)$
  Storage: $O(n/\varepsilon^{d/2})$

Space-Time Tradeoffs for ANN



(a) First generation (before 2002)

(b) AVDs [AMM09]

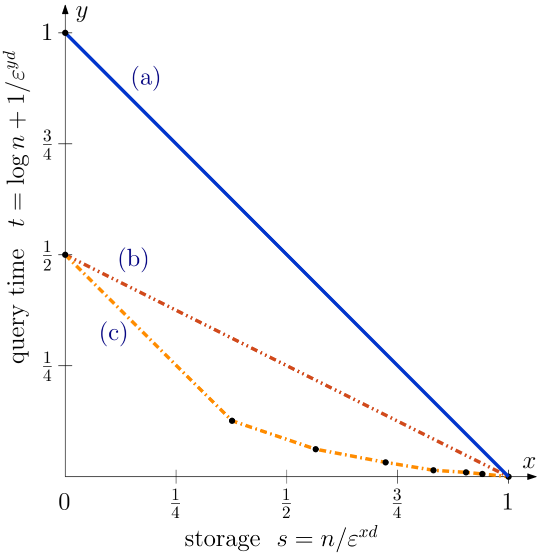
(c) Reduction to Split-Reduce

(d) Reduction to Macbeath regions

Best Upper Bound

- For $\log \frac{1}{\varepsilon} \leq m \leq 1/\varepsilon^{d/2}$
  Query time: $O(\log n + 1/(m\, \varepsilon^{d/2}))$
  Storage: $O(n\, m)$
- Setting $m = 1/\varepsilon^{d/2}$
  Query time: $O(\log n)$
  Storage: $O(n/\varepsilon^{d/2})$

Space-Time Tradeoffs for ANN



(a) First generation (before 2002)

(b) AVDs [AMM09]

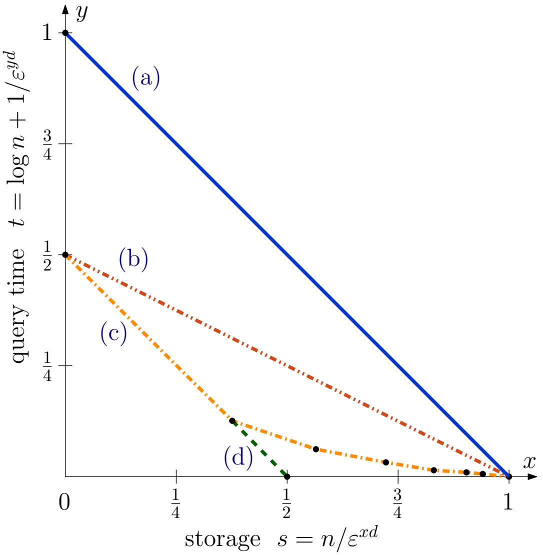
(c) Reduction to Split-Reduce

(d) Reduction to Macbeath regions

Best Upper Bound

- For $\log \frac{1}{\varepsilon} \leq m \leq 1/\varepsilon^{d/2}$
  Query time: $O(\log n + 1/(m\,\varepsilon^{d/2}))$
  Storage: $O(n\,m)$
- Setting $m = 1/\varepsilon^{d/2}$
  Query time: $O(\log n)$
  Storage: $O(n/\varepsilon^{d/2})$

Space-Time Tradeoffs for ANN



(a) First generation (before 2002)

(b) AVDs [AMM09]

(c) Reduction to Split-Reduce

(d) Reduction to Macbeath regions

Best Upper Bound

- For $\log \frac{1}{\varepsilon} \leq m \leq 1/\varepsilon^{d/2}$
  Query time: $O(\log n + 1/(m\, \varepsilon^{d/2}))$
  Storage: $O(n\, m)$
- Setting $m = 1/\varepsilon^{d/2}$
  Query time: $O(\log n)$
  Storage: $O(n/\varepsilon^{d/2})$

Directional Width

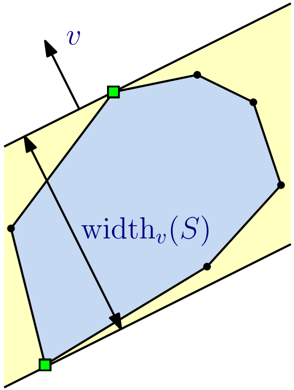
Directional width

Given:

- S: set of n points in $\mathbb{R}^d$
- v: unit vector

Define $\mathrm{width}_v(S)$:

- Minimum distance between two hypeplanes orthogonal to v enclosing S



ε-Kernel

Input

S: Set of n points in $\mathbb{R}^d$

$\varepsilon > 0$: Approximation parameter

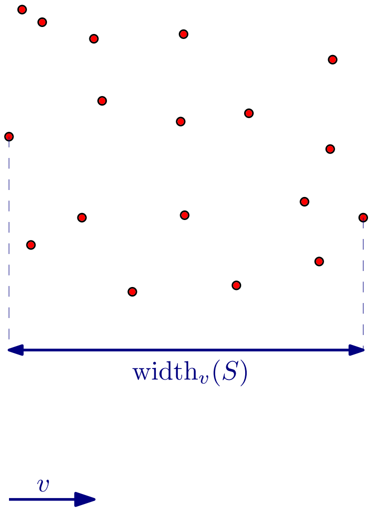
Output

$Q \subseteq S$ such that for all vector v,

$$\text{width}_v(Q) \geq (1-\varepsilon)\,\text{width}_v(S)$$

and $|Q| = O(1/\varepsilon^{(d-1)/2})$

- Approximation of the convex hull
- Minimum size: $\Theta(1/\varepsilon^{(d-1)/2})$



ε-Kernel

Input

S: Set of n points in $\mathbb{R}^d$

$\varepsilon > 0$: Approximation parameter

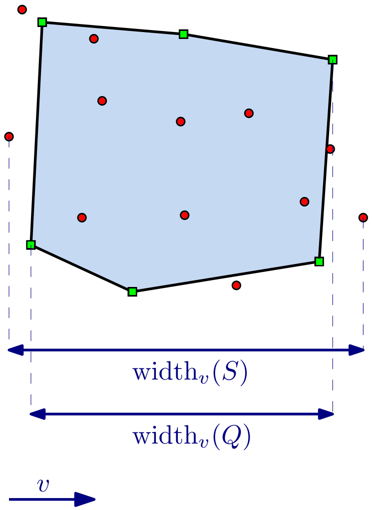
Output

$Q \subseteq S$ such that for all vector v,

$$\text{width}_v(Q) \geq (1-\varepsilon)\ \text{width}_v(S)$$

and $|Q| = O(1/\varepsilon^{(d-1)/2})$

- Approximation of the convex hull
- Minimum size: $\Theta(1/\varepsilon^{(d-1)/2})$

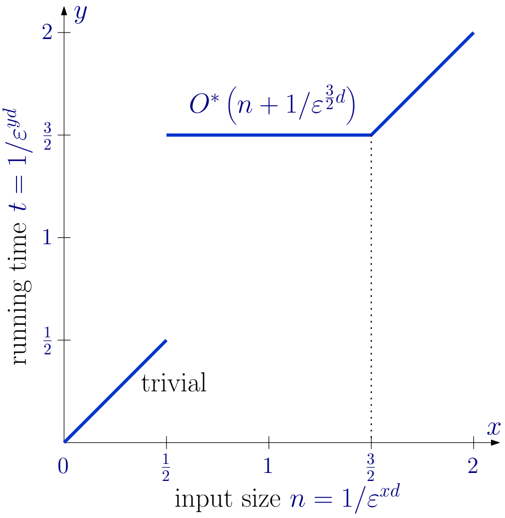


History of ε-Kernel Algorithms

- [AHV04] $O\left(n + 1/\varepsilon^{\frac{3(d-1)}{2}}\right)$
- [Cha06] $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-2}\right)$
- [ArC14] $O\left(n + \sqrt{n}/\varepsilon^{\frac{d}{2}}\right)$
- [Cha17] $\widetilde{O}\left(n\sqrt{\frac{1}{\varepsilon}} + 1/\varepsilon^{\frac{d-1}{2}+\frac{3}{2}}\right)$

Our near-optimal construction

- $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$
- $\alpha > 0$ arbitrarily small
- Independent of [Cha17] and completely different technique

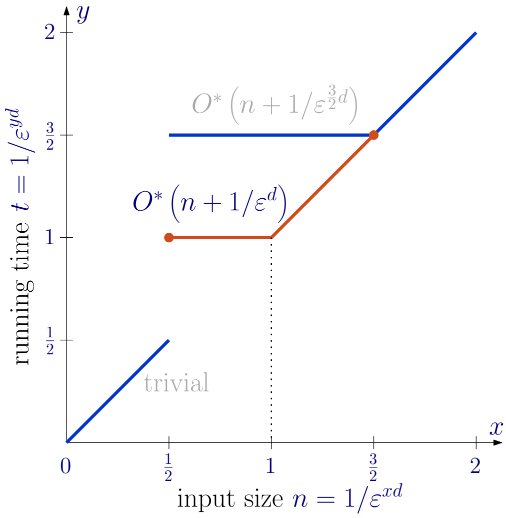


History of ε-Kernel Algorithms

- [AHV04] $O\left(n + 1/\varepsilon^{\frac{3(d-1)}{2}}\right)$
- [Cha06] $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-2}\right)$
- [ArC14] $O\left(n + \sqrt{n}/\varepsilon^{\frac{d}{2}}\right)$
- [Cha17] $\widetilde{O}\left(n\sqrt{\frac{1}{\varepsilon}} + 1/\varepsilon^{\frac{d-1}{2}+\frac{3}{2}}\right)$

Our near-optimal construction

- $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$
- $\alpha > 0$ arbitrarily small
- Independent of [Cha17] and completely different technique

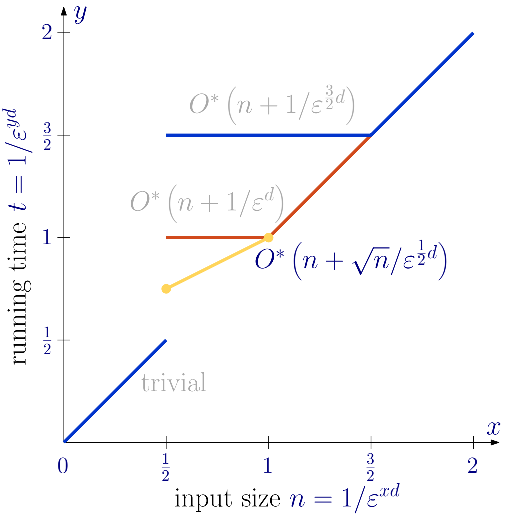


History of ε-Kernel Algorithms

- [AHV04] $O\left(n + 1/\varepsilon^{\frac{3(d-1)}{2}}\right)$
- [Cha06] $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-2}\right)$
- [ArC14] $O\left(n + \sqrt{n}/\varepsilon^{\frac{d}{2}}\right)$
- [Cha17] $\widetilde{O}\left(n\sqrt{\frac{1}{\varepsilon}} + 1/\varepsilon^{\frac{d-1}{2}+\frac{3}{2}}\right)$

Our near-optimal construction

- $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$
- $\alpha > 0$ arbitrarily small
- Independent of [Cha17] and completely different technique

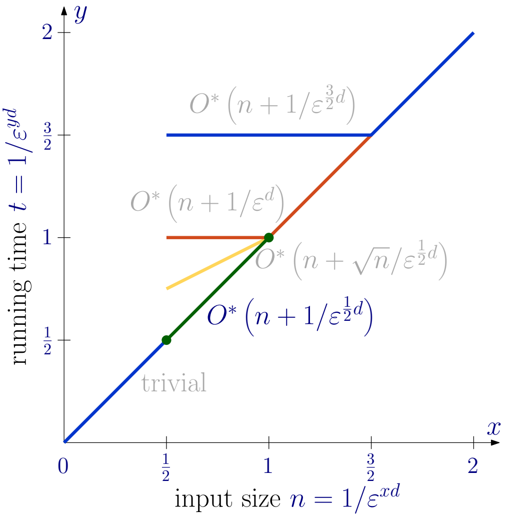


History of ε-Kernel Algorithms

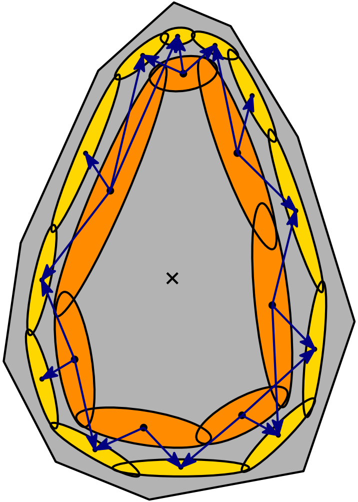
- [AHV04] $O\left(n + 1/\varepsilon^{\frac{3(d-1)}{2}}\right)$
- [Cha06] $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-2}\right)$
- [ArC14] $O\left(n + \sqrt{n}/\varepsilon^{\frac{d}{2}}\right)$
- [Cha17] $\widetilde{O}\left(n\sqrt{\frac{1}{\varepsilon}} + 1/\varepsilon^{\frac{d-1}{2}+\frac{3}{2}}\right)$

Our near-optimal construction

- $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$
- $\alpha > 0$ arbitrarily small
- Independent of [Cha17] and completely different technique



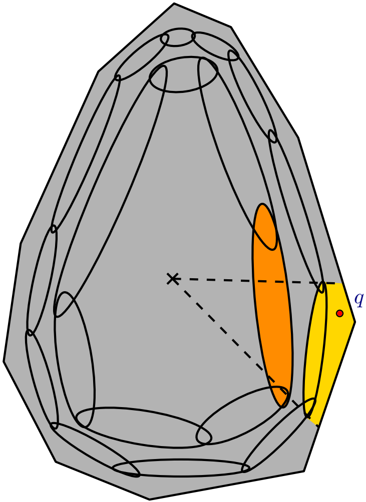
Hierarchy of Macbeath Ellipsoids



- Hierarchy construction takes: $O\left(n + 1/\varepsilon^{\frac{3(d-1)}{2}}\right)$ time
- Input polytope may be described as:
  - Intersection of n halfspaces
  - Convex hull of n points
- Too slow to efficiently build ε-kernel

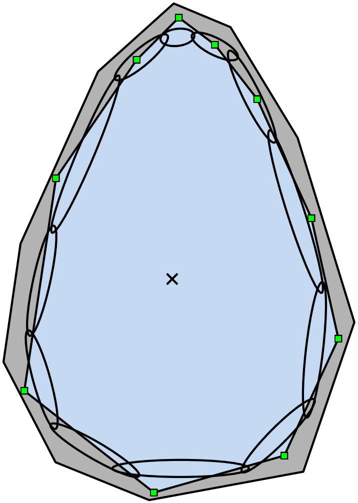
Hierarchy Properties

- Query point $q \in K$:
  - Find leaf shadow that contains q
  - Or report q as far from the boundary
  - $O(\log \frac{1}{\varepsilon})$ time
- Hierarchy $\longrightarrow$ Kernel
  - Split points among leaf shadows
  - Pick one point per leaf shadow (if there's one)
  - $O(n \log \frac{1}{\varepsilon})$ time

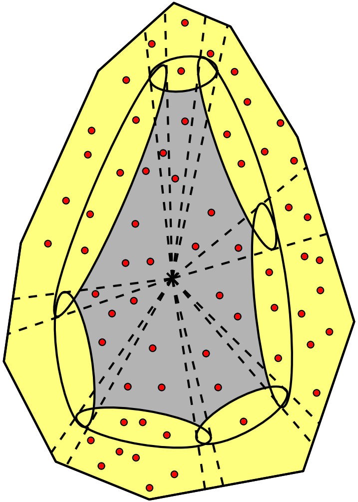


Hierarchy Properties

- Query point $q \in K$:
  - Find leaf shadow that contains q
  - Or report q as far from the boundary
  - $O(\log \frac{1}{\varepsilon})$ time
- Hierarchy $\longrightarrow$ Kernel
  - Split points among leaf shadows
  - Pick one point per leaf shadow (if there's one)
  - $O(n \log \frac{1}{\varepsilon})$ time



Kernel Construction [AFM17b]

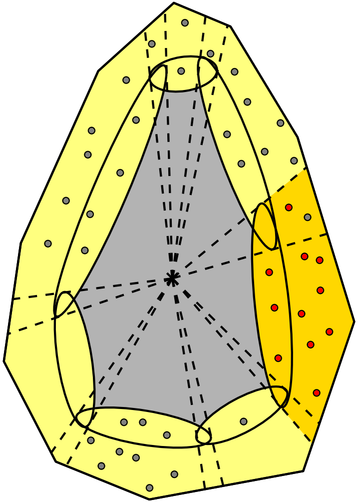


1. Build hierarchy for $\delta = \varepsilon^{1/3}$:
   $O\left(n + 1/\delta^{\frac{3(d-1)}{2}}\right) = O\left(n + 1/\varepsilon^{\frac{d-1}{2}}\right)$ time
2. Split points among shadows: $O(n \log \frac{1}{\varepsilon})$ time
3. Build $\frac{\varepsilon}{\delta}$-kernel for each shadow
   (using existing $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-1})$ algorithm)
   $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{d-1}\right) =$
   $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{5(d-1)}{6}}\right)$
4. Return union of kernels

Time: $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{5(d-1)}{6}}\right)$

Kernel size: $O\left(\left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{\frac{d-1}{2}}\right) = O\left(1/\varepsilon^{\frac{d-1}{2}}\right)$

Kernel Construction [AFM17b]

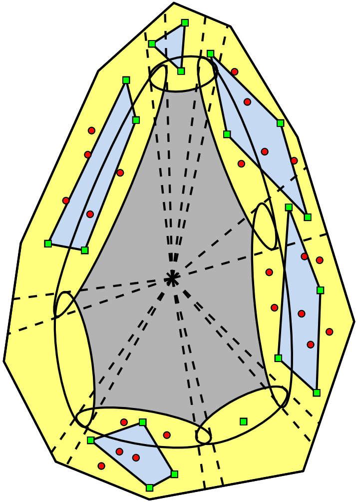


1. Build hierarchy for $\delta = \varepsilon^{1/3}$:
   $O\left(n + 1/\delta^{\frac{3(d-1)}{2}}\right) = O\left(n + 1/\varepsilon^{\frac{d-1}{2}}\right)$ time
2. Split points among shadows: $O(n \log \frac{1}{\varepsilon})$ time
3. Build $\frac{\varepsilon}{\delta}$-kernel for each shadow (using existing $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-1})$ algorithm)
   $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{d-1}\right) =$
   $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{5(d-1)}{6}}\right)$
4. Return union of kernels

Time: $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{5(d-1)}{6}}\right)$

Kernel size: $O\left(\left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{\frac{d-1}{2}}\right) = O\left(1/\varepsilon^{\frac{d-1}{2}}\right)$

Kernel Construction [AFM17b]

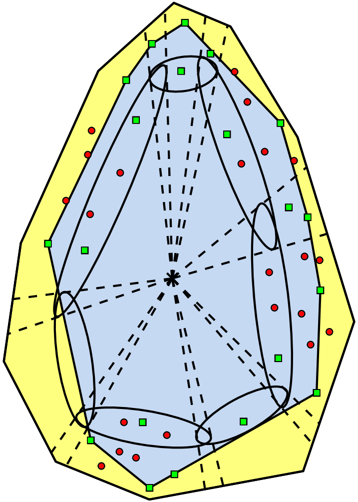


1. Build hierarchy for $\delta = \varepsilon^{1/3}$:
   $O\left(n + 1/\delta^{\frac{3(d-1)}{2}}\right) = O\left(n + 1/\varepsilon^{\frac{d-1}{2}}\right)$ time
2. Split points among shadows: $O(n \log \frac{1}{\varepsilon})$ time
3. Build $\frac{\varepsilon}{\delta}$-kernel for each shadow
   (using existing $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-1})$ algorithm)
   $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{d-1}\right) =$
   $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{5(d-1)}{6}}\right)$
4. Return union of kernels

Time: $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{5(d-1)}{6}}\right)$

Kernel size: $O\left(\left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{\frac{d-1}{2}}\right) = O\left(1/\varepsilon^{\frac{d-1}{2}}\right)$

Kernel Construction [AFM17b]

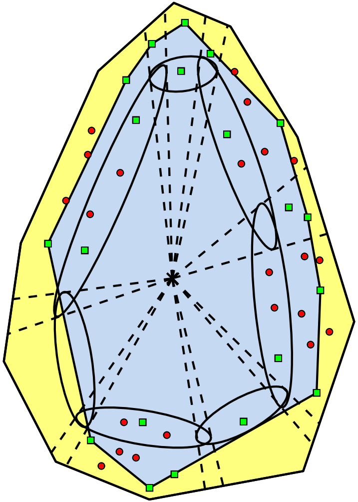


1. Build hierarchy for $\delta = \varepsilon^{1/3}$:
   $O\left(n + 1/\delta^{\frac{3(d-1)}{2}}\right) = O\left(n + 1/\varepsilon^{\frac{d-1}{2}}\right)$ time
2. Split points among shadows: $O(n \log \frac{1}{\varepsilon})$ time
3. Build $\frac{\varepsilon}{\delta}$-kernel for each shadow
   (using existing $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-1})$ algorithm)
   $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{d-1}\right) =$
   $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{5(d-1)}{6}}\right)$
4. Return union of kernels

Time: $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{5(d-1)}{6}}\right)$

Kernel size: $O\left(\left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{\frac{d-1}{2}}\right) = O\left(1/\varepsilon^{\frac{d-1}{2}}\right)$

Kernel Construction [AFM17b]



1. Build hierarchy for $\delta = \varepsilon^{1/3}$:
   $O\left(n + 1/\delta^{\frac{3(d-1)}{2}}\right) = O\left(n + 1/\varepsilon^{\frac{d-1}{2}}\right)$ time
2. Split points among shadows: $O(n \log \frac{1}{\varepsilon})$ time
3. Build $\frac{\varepsilon}{\delta}$-kernel for each shadow
   (using existing $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{d-1})$ algorithm)
   $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{d-1}\right) =$
   $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{5(d-1)}{6}}\right)$
4. Return union of kernels

Time: $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{5(d-1)}{6}}\right)$

Kernel size: $O\left(\left(\frac{1}{\delta}\right)^{\frac{d-1}{2}} \left(\frac{\delta}{\varepsilon}\right)^{\frac{d-1}{2}}\right) = O\left(1/\varepsilon^{\frac{d-1}{2}}\right)$

Bootstrapping

Bootstrap using improved ε-kernel construction:

- $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{t(d-1)}\right)$ time $\longrightarrow O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{4t+1}{6}(d-1)}\right)$ time
- $t: \; 1 \longrightarrow \frac{5}{6} \longrightarrow \frac{13}{18} \longrightarrow \frac{35}{54} \longrightarrow \cdots \longrightarrow \frac{1}{2} + \alpha$
- Exponent t arbitrarily close to $\frac{1}{2}$

Running Time

$O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$, for arbitrarily small $\alpha > 0$

Bootstrapping

Bootstrap using improved ε-kernel construction:

- $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{t(d-1)}\right)$ time $\longrightarrow O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{4t+1}{6}(d-1)}\right)$ time
- $t: \; 1 \longrightarrow \frac{5}{6} \longrightarrow \frac{13}{18} \longrightarrow \frac{35}{54} \longrightarrow \cdots \longrightarrow \frac{1}{2} + \alpha$
- Exponent t arbitrarily close to $\frac{1}{2}$

Running Time

$O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$, for arbitrarily small $\alpha > 0$

Bootstrapping

Bootstrap using improved ε-kernel construction:

- $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{t(d-1)}\right)$ time $\longrightarrow O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{4t+1}{6}(d-1)}\right)$ time
- $t: \; 1 \longrightarrow \frac{5}{6} \longrightarrow \frac{13}{18} \longrightarrow \frac{35}{54} \longrightarrow \cdots \longrightarrow \frac{1}{2} + \alpha$
- Exponent t arbitrarily close to $\frac{1}{2}$

Running Time

$O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$, for arbitrarily small $\alpha > 0$

Bootstrapping

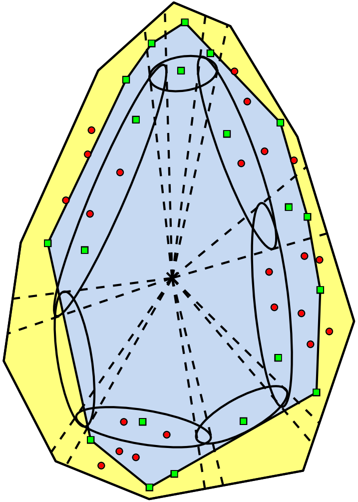
Bootstrap using improved ε-kernel construction:

- $O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{t(d-1)}\right)$ time $\longrightarrow O\left(n \log \frac{1}{\varepsilon} + \left(\frac{1}{\varepsilon}\right)^{\frac{4t+1}{6}(d-1)}\right)$ time
- $t:\ 1 \longrightarrow \frac{5}{6} \longrightarrow \frac{13}{18} \longrightarrow \frac{35}{54} \longrightarrow \cdots \longrightarrow \frac{1}{2} + \alpha$
- Exponent t arbitrarily close to $\frac{1}{2}$

Running Time

$O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$, for arbitrarily small $\alpha > 0$

Preprocessing Approximate Polytope Membership



- Same strategy to efficiently preprocess an approximate polytope membership data structure

Approximate Polytope Membership

- Query time: $O(\log \frac{1}{\varepsilon})$ ← optimal
- Storage: $O(1/\varepsilon^{\frac{d-1}{2}})$ ← optimal
- Preprocessing: $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha})$
  ↑ almost optimal

Approximate Diameter [AFM17b]

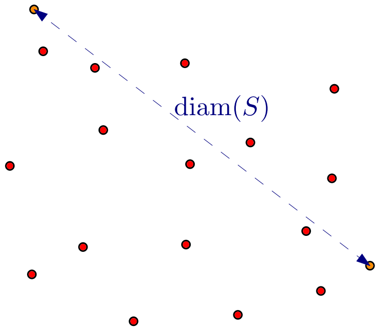
Input

S: Set of n points in $\mathbb{R}^d$
$\varepsilon > 0$: Approximation parameter

Output

$p, q \in S$ with

$$\|pq\| \geq (1 - \varepsilon)\,\mathrm{diam}(S)$$



Approximate Diameter [AFM17b]

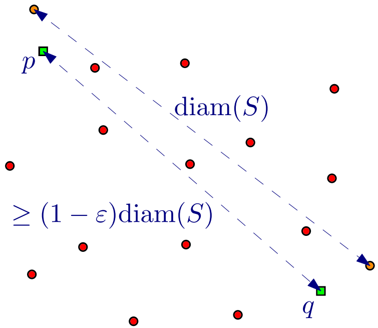
Input

S: Set of n points in $\mathbb{R}^d$
$\varepsilon > 0$: Approximation parameter

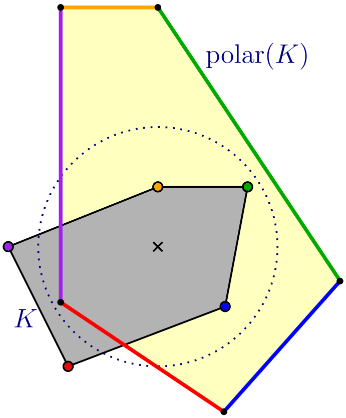
Output

$p, q \in S$ with

$$\|pq\| \geq (1 - \varepsilon)\ \mathrm{diam}(S)$$

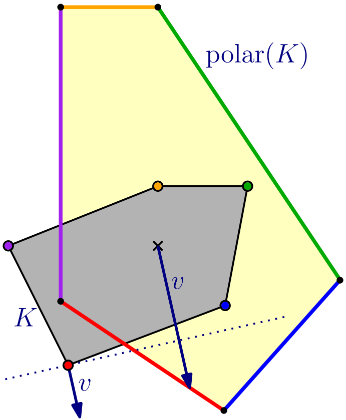


Polarity



- K: convex body
- Polar of K:
  points p such that $p \cdot q \leq 1$ for $q \in K$
- In K: extreme point in direction v
- In $\text{polar}(K)$: ray shooting in direction v from origin

Polarity

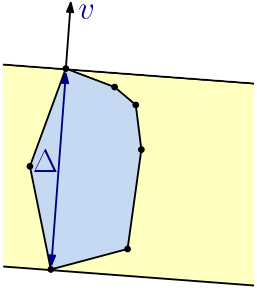


- K: convex body
- Polar of K:
  points p such that $p \cdot q \leq 1$ for $q \in K$
- In K: extreme point in direction v
- In $\text{polar}(K)$: ray shooting in direction v from origin

Diameter by Extreme Points

- Diameter: $\max_v \mathrm{width}_v(K)$
- Diameter: Approximated using $O(1/\varepsilon^{\frac{d-1}{2}})$ directional width queries [Cha02]

1. Preprocess $\mathrm{polar}(K)$ for ray shooting
2. Perform $O(1/\varepsilon^{\frac{d-1}{2}})$ directional width queries on K
3. Return maximum width found



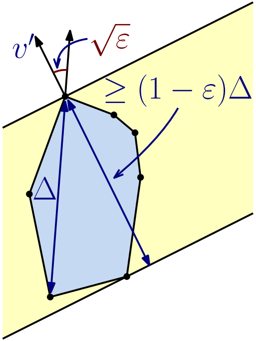
Running Time

$O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$, for arbitrarily small $\alpha > 0$

Diameter by Extreme Points

- Diameter: $\max_v \operatorname{width}_v(K)$
- Diameter: Approximated using $O(1/\varepsilon^{\frac{d-1}{2}})$ directional width queries [Cha02]

1. Preprocess $\operatorname{polar}(K)$ for ray shooting
2. Perform $O(1/\varepsilon^{\frac{d-1}{2}})$ directional width queries on K
3. Return maximum width found



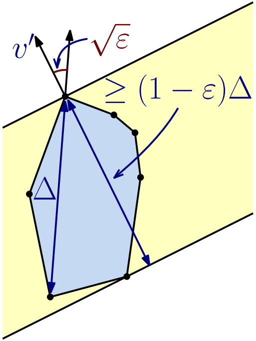
Running Time

$O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$, for arbitrarily small $\alpha > 0$

Diameter by Extreme Points

- Diameter: $\max_v \text{width}_v(K)$
- Diameter: Approximated using $O(1/\varepsilon^{\frac{d-1}{2}})$ directional width queries [Cha02]

1. Preprocess $\text{polar}(K)$ for ray shooting
2. Perform $O(1/\varepsilon^{\frac{d-1}{2}})$ directional width queries on K
3. Return maximum width found



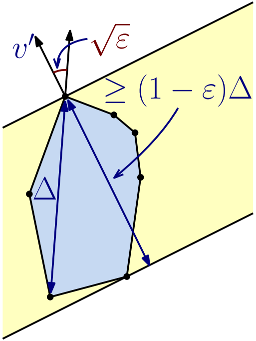
Running Time

$O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$, for arbitrarily small $\alpha > 0$

Diameter by Extreme Points

- Diameter: $\max_v \text{width}_v(K)$
- Diameter: Approximated using $O(1/\varepsilon^{\frac{d-1}{2}})$ directional width queries [Cha02]

1. Preprocess $\text{polar}(K)$ for ray shooting
2. Perform $O(1/\varepsilon^{\frac{d-1}{2}})$ directional width queries on K
3. Return maximum width found



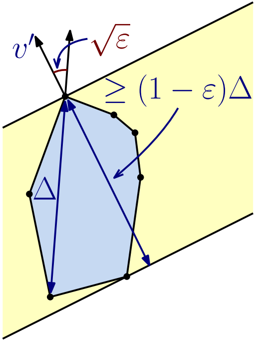
Running Time

$O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$, for arbitrarily small $\alpha > 0$

Diameter by Extreme Points

- Diameter: $\max_v \mathrm{width}_v(K)$
- Diameter: Approximated using $O(1/\varepsilon^{\frac{d-1}{2}})$ directional width queries [Cha02]

1. Preprocess $\mathrm{polar}(K)$ for ray shooting
2. Perform $O(1/\varepsilon^{\frac{d-1}{2}})$ directional width queries on K
3. Return maximum width found



Running Time

$O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$, for arbitrarily small $\alpha > 0$

Results

Our approximate polytope membership data structure is optimal

- Query time: $O(\log \frac{1}{\varepsilon})$
- Storage: $O(1/\varepsilon^{\frac{d-1}{2}})$
- Preprocessing: $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha})$

We showed how to use it to obtain:

- ANN searching in $O(\log n)$ query time with $O(n/\varepsilon^{d/2})$ storage
- Near-optimal ε-kernel construction in $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$ time
- Diameter approximation in $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$ time
- Bichromatic closest pair approximation in $O\left(n/\varepsilon^{\frac{d}{4}+\alpha}\right)$ expected time
- Euclidean minimum spanning/bottleneck tree approximation in $O\left((n \log n)/\varepsilon^{\frac{d}{4}+\alpha}\right)$ expected time

Results

Our approximate polytope membership data structure is optimal

- Query time: $O(\log \frac{1}{\varepsilon})$
- Storage: $O(1/\varepsilon^{\frac{d-1}{2}})$
- Preprocessing: $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha})$

We showed how to use it to obtain:

- ANN searching in $O(\log n)$ query time with $O(n/\varepsilon^{d/2})$ storage
- Near-optimal ε-kernel construction in $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$ time
- Diameter approximation in $O\left(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{\frac{d-1}{2}+\alpha}\right)$ time
- Bichromatic closest pair approximation in $O\left(n/\varepsilon^{\frac{d}{4}+\alpha}\right)$ expected time
- Euclidean minimum spanning/bottleneck tree approximation in $O\left((n \log n)/\varepsilon^{\frac{d}{4}+\alpha}\right)$ expected time

Open Problems

Still, several open problems remain

- Faster preprocessing
- Further improvements to approximate nearest neighbor searching
- Generalization to k-nearest neighbors
- Lower bound for diameter (or improved upper bound)
- Diameter for non-Euclidean metrics
- Other applications of the hierarchy

Ongoing work:

- Approximate the width
- Approximate polytope intersection
- ANN with non-Euclidean metrics

References

[AHV04] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. Assoc. Comput. Mach.*, 51:606–635, 2004.

[ArC14] S. Arya and T. M. Chan. Better ε-dependencies for offline approximate nearest neighbor search, Euclidean minimum spanning trees, and ε-kernels. In *Proc. 30th Sympos. Comput. Geom.*, pages 416–425, 2014.

[AFM17a] S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal approximate polytope membership. In *Proc. 28th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 270–288, 2017.

[AFM17b] S. Arya, G. D. da Fonseca, and D. M. Mount. Near-optimal ε-kernel construction and related problems. In *Proc. 33rd Internat. Sympos. Comput. Geom.*, pages 10:1–15, 2017.

[AFM17c] S. Arya, G. D. da Fonseca, and D. M. Mount. On the combinatorial complexity of approximating polytopes. *Discrete Comput. Geom.*, 58(4):849–870, 2017.

[AFM18] S. Arya, G. D. da Fonseca, and D. M. Mount. Approximate polytope membership queries. *SIAM J. Comput.*, 47(1):1–51, 2018.

[AMM09] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. Assoc. Comput. Mach.*, 57:1–54, 2009.

[Bar07] I. Bárány. Random polytopes, convex bodies, and approximation. In W. Weil, editor, *Stochastic Geometry*, volume 1892 of *Lecture Notes in Mathematics*, pages 77–118, 2007.

[Cha02] T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. *Internat. J. Comput. Geom. Appl.*, 12:67–85, 2002.

[Cha06] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35(1):20–35, 2006.

[Cha17] T. M. Chan. Applications of Chebyshev polynomials to low-dimensional computational geometry. In *Proc. 33rd Internat. Sympos. Comput. Geom.*, 2017.

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

[Joh48] F. John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday*, pages 187–204, 1948.

[Mac52] A. M. Macbeath. A theorem on non-homogeneous lattices. *Annals of Mathematics*, 54:431–438, 1952.

Painting by Robert Delaunay

Thank you!