CS 372: Computational Geometry
Lecture 14
Geometric Approximation Algorithms

Antoine Vigneron

King Abdullah University of Science and Technology

December 5, 2012

Outline

Introduction to geometric approximation algorithms.

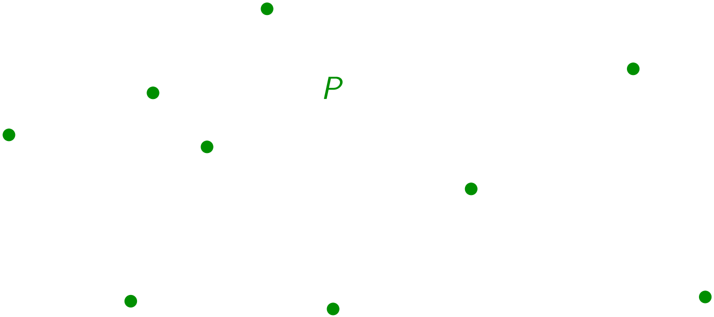
Example: Computing the diameter of a point-set.

- Simple 2-approximation algorithm.
- Two approaches for ε-approximation:
  - Rounding.
  - Projection.

References:

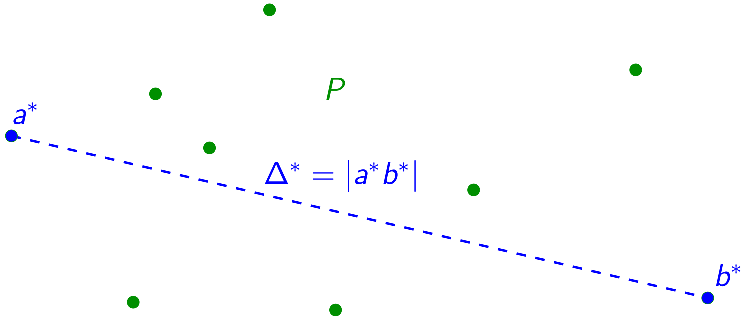
- Sariel Har Peled's book.
- Paper by T. Chan, *Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus*, Section 2.

The Diameter Problem



- Input: a set P of n points in $\mathbb{R}^d$.

The Diameter Problem



- Input: a set P of n points in $\mathbb{R}^d$.
- Output: the maximum distance Δ^* between any two points of P.
- Δ^* is called the *diameter* of P.

Brute Force Algorithm

Computing the distance between two points:

- If $a = (a_1, a_2, \ldots, a_d)$ and $b = (b_1, b_2, \ldots, b_d)$, then

$$|ab| = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + \cdots + (b_d - a_d)^2}.$$

- It takes $O(d)$ time.
- Here we assume that d is constant: $d = O(1)$.
  - We are in *fixed dimension*.
- Then we can compute $|ab|$ in $O(1)$ time.

Computing the diameter:

- Brute force: Check all pairs in P^2 and keep the maximum distance.
- Brute force takes $O(n^2)$ time.

Approximation Algorithms

Definition (c-factor approximation)

We say that Δ is a c-factor approximation to Δ^* if $\Delta \leqslant \Delta^* \leqslant c\Delta$.

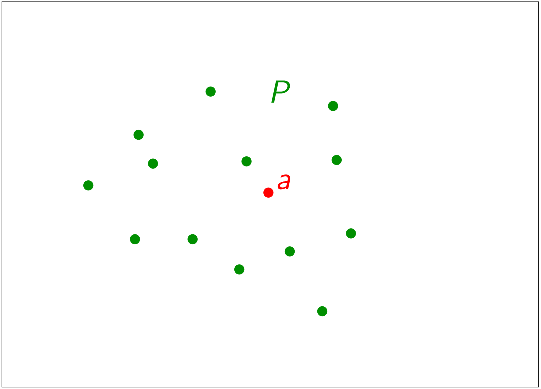
Definition (c-approximation algorithm)

A c-approximation algorithm for a maximization problem is an algorithm that computes a c-factor approximation of the optimum in polynomial time.

We will give a 2-approximation algorithm for the diameter problem.

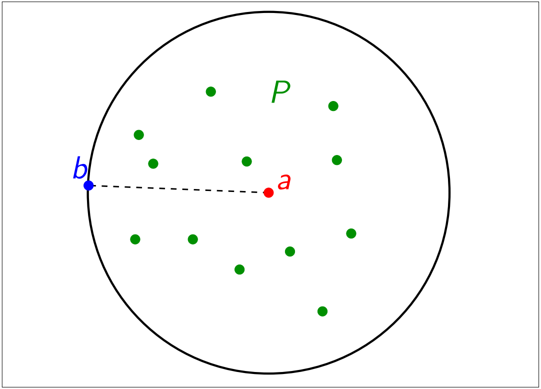
- That is, we find Δ_0 such that $\Delta_0 \leqslant \Delta^* \leqslant 2\Delta_0$
- $O(n)$ time, very simple.
- Best known exact algorithms are slower and complicated.

2-Approximation Algorithm



- Pick a point $a \in P$.

2-Approximation Algorithm



- Pick a point $a \in P$.
- Find b such that $|ab|$ is maximum.
- $\Delta_0 = |ab|$.

Analysis

- Pick any point $a \in P$.
  - $O(1)$ time.
- Find $b \in P$ such that $|ab|$ is maximum.
  - Just go through the list of points in P and keep the maximum distance to a.
  - It takes $O(n)$ time.
- Conclusion: This algorithm runs in $O(n)$ time.

Proof of Correctness

- We need to prove that Δ_0 is a 2-factor approximation. of Δ^*.
- It means $\Delta_0 \leqslant \Delta^* \leqslant 2\Delta_0$.
- Since $(a, b) \in P^2$, clearly, $\Delta_0 = |ab| \leqslant \Delta^*$.
- We also need to prove that $\Delta^* \leqslant 2\Delta_0$.
  - Let a^* and b^* denote two points such that $|a^* b^*| = \Delta^*$.
  - By definition of b we have $|aa^*| \leqslant |ab|$ and $|ab^*| \leqslant |ab|$.
  - By the triangle inequality

$$
\begin{aligned}
\Delta^* &= |a^* b^*| \\
&\leqslant |a^* a| + |ab^*| \\
&\leqslant |ab| + |ab| \\
&= 2\Delta_0.
\end{aligned}
$$

Concluding Remark

- The running time is optimal.
- If we do not assume $d = O(1)$:
  - This algorithm still works.
  - But the running time has to be written $O(dn)$.
  - It is still optimal as we need $\Theta(nd)$ time to read the input.

$(1+\varepsilon)$-Approximation Algorithms

- We just found a 2-approximation algorithm.
- We would like to obtain a better approximation.
- Let $\varepsilon > 0$ be a real number.
  - In this lecture, we assume $\varepsilon < 1$.
  - ε should be thought of as being small, say $\varepsilon = 0.1$ or $\varepsilon = 0.01$.
- We want to design a $(1+\varepsilon)$-approximation algorithm.
- The result will be Δ such that $\Delta \leqslant \Delta^* \leqslant (1+\varepsilon)\Delta$.
  - In other words, the *relative error* we allow is ε.
  - So $\varepsilon = 0.01$ means a 1% error.

Analysis

- How to analyze a $(1+\varepsilon)$-approximation algorithm?
- The running time is expressed as a function of n and ε using $O(\cdot)$ notation.
  - For instance, the last algorithm in this lecture runs in time.

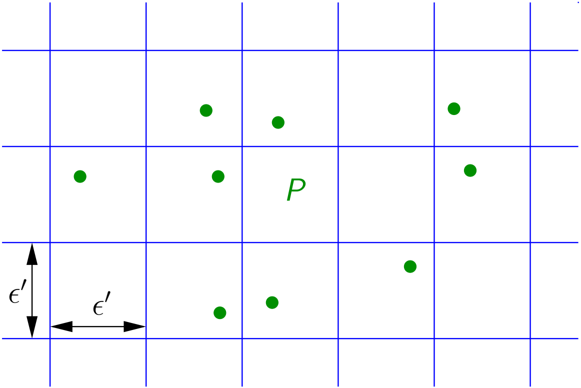
$$O\left(n + (1/\varepsilon)^{\frac{3}{2}(d-1)}\right)$$

  - Since $d = 0(1)$, it is polynomial in n and $1/\varepsilon$.
    - ⋆ It is an *FPTAS* (Fully Polynomial Time Approximation Scheme).
- The running time is linear in n.
- Problem: exponential in d.
  - These algorithms are only interesting in low dimension d.

Idea

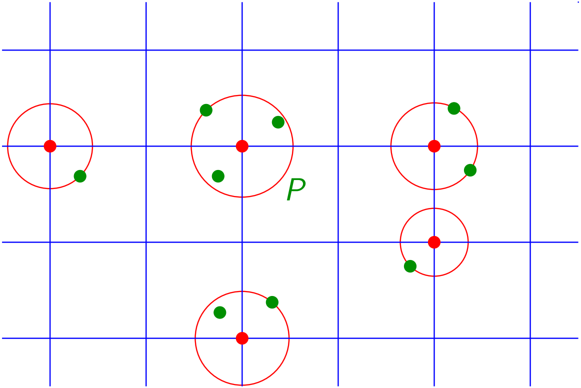
- Start with a set P of n points.
- Transform it into a set P' such that:
  - The cardinality $|P'|$ is small.
  - The diameter Δ' of P' is a good approximation of Δ.
- Then find Δ' by brute force

Rounding to a Grid



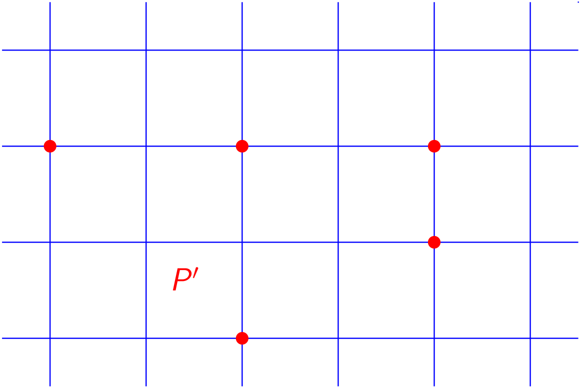
- Consider a regular grid over $\mathbb{R}^d$.
- The side length of the grid is ϵ', to be specified later.
- Intuition: we will choose $\epsilon' \simeq \epsilon\Delta^*$, it is the error we allow.

Rounding to a Grid



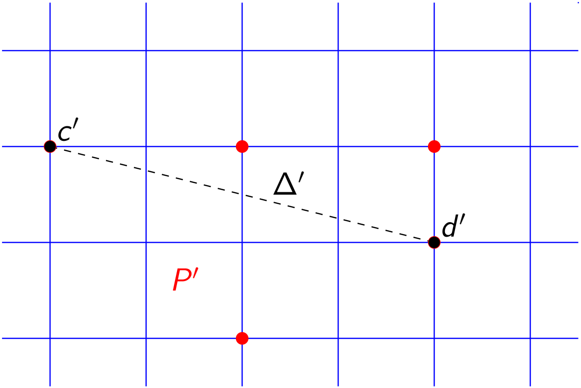
- Replace each point of P with the nearest grid point.
- This operation is called *rounding*.

Rounding to a Grid



- The grid points we obtain form the set P'.

Rounding to a Grid



- Compute the diameter Δ' of P' by brute force.

Intuition

- P' is a d-dimensional point set with diameter Δ'.
- The points are on a grid with side length ϵ'; we will choose it such that $\epsilon' \simeq \Delta'\epsilon$.
- So in the worst case, there are about as many points in P' as in a $(1/\epsilon) \times (1/\epsilon) \cdots \times (1/\epsilon)$ grid in $\mathbb{R}^d$.
- There are $O((1/\epsilon)^d)$ such points.
- We can compute Δ' by brute force in time $O((1/\epsilon)^{2d})$.

How to Perform Rounding?

- The grid points have coordinates $(k_1\epsilon', k_2\epsilon', \dots k_d\epsilon')$ where each k_i is an integer.
- Let $p = (p_1, p_2, \dots p_d) \in P$.
- How can we find the closest grid point p'?
- We need to find the closest integer k_i to p_i/ϵ'.
- It is given by the formula

$$k_i = \left\lfloor \frac{p_i}{\epsilon'} + \frac{1}{2} \right\rfloor .$$

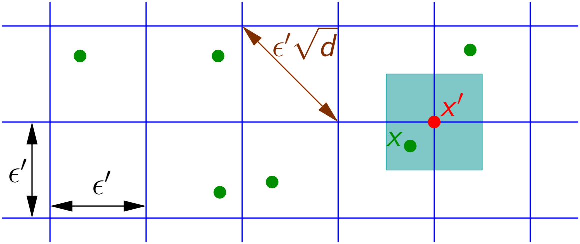
- p is rounded to $p' = (k_1\epsilon', k_2\epsilon', \dots k_d\epsilon')$.
- It takes $O(d) = O(1)$ time.

Rounding Error

Property

Let $x \in \mathbb{R}^d$, and let x' be the closest grid point to x. Then $|xx'| \leqslant \epsilon' \sqrt{d}/2$.

- Proof: Apply previous slide formula.
- Intuition:
  - x is in a hypercube centered at x' with side length ϵ'.



  - This hypercube diagonal has length $\epsilon' \sqrt{d}$.

Approximation Factor

- Let a' and b' be the closest grid points to a^* and b^*, respectively.
- Then from previous slide, $|a'a^*| \leqslant \epsilon'\sqrt{d}/2$ and $|b'b^*| \leqslant \epsilon'\sqrt{d}/2$.
- By the triangle inequality,

$$\begin{aligned}\Delta^* &= |a^*b^*| \\ &\leqslant |a^*a'| + |a'b'| + |b'b^*| \\ &\leqslant |a'b'| + \epsilon'\sqrt{d} \\ &\leqslant \Delta' + \epsilon'\sqrt{d}.\end{aligned}$$

Approximation Factor

- Let c' and d' be two points of P' such that $|c'd'| = \Delta'$.
- Let c and d be points of P that have been rounded to c' and d', respectively.
- Then $|cc'| \leqslant \epsilon' \sqrt{d}/2$ and $|dd'| \leqslant \epsilon' \sqrt{d}/2$.
- So by the triangle inequality,

$$\begin{aligned} \Delta' &= |c'd'| \\ &\leqslant |cd| + \epsilon' \sqrt{d}. \end{aligned}$$

- Of course $|cd| \leqslant \Delta^*$.
- It follows that

$$\Delta' \leqslant \Delta^* + \epsilon' \sqrt{d}$$

Approximation Factor

- We obtained
$$\Delta' - \epsilon'\sqrt{d} \leqslant \Delta^* \leqslant \Delta' + \epsilon'\sqrt{d}.$$
- We choose $\Delta = \Delta' - \epsilon'\sqrt{d}$, so
$$\Delta \leqslant \Delta^* \leqslant \Delta + 2\epsilon'\sqrt{d}.$$
- We want to make sure that $2\epsilon'\sqrt{d} \leqslant \epsilon\Delta^*/2$.
- Since $\Delta_0 \leqslant \Delta^*$, it suffices that $2\epsilon'\sqrt{d} \leqslant \epsilon\Delta_0/2$.
- So we first compute Δ_0, and then we set $\epsilon' = \epsilon\Delta_0/(4\sqrt{d})$.
- It follows that $\Delta \leqslant \Delta^* \leqslant \Delta + \epsilon\Delta^*/2$.
- Problem: We wanted to show that $\Delta \leqslant \Delta^* \leqslant \Delta + \epsilon\Delta$.

Technical Difficulty

- Intuition: The relative error is less than ϵ, so it is OK.
- Proof: We want to prove that $\Delta^* \leqslant \Delta + \epsilon\Delta$.
  - We know that $\Delta^* \leqslant \Delta + \epsilon\Delta^*/2$.
  - Then

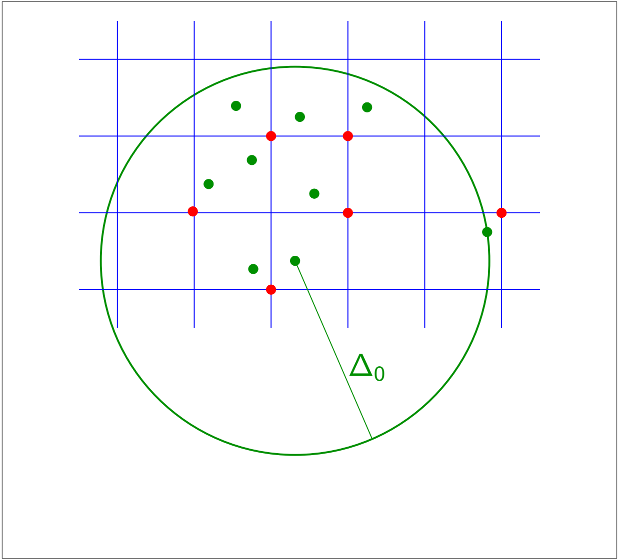
$$\begin{aligned}\Delta^* &\leqslant \frac{1}{1-\epsilon/2}\Delta \\ &= \left(1 + \frac{\epsilon}{2} + \frac{\epsilon^2}{4} + \frac{\epsilon^3}{8} + \dots\right)\Delta.\end{aligned}$$

  - Since $\epsilon < 1$, we get

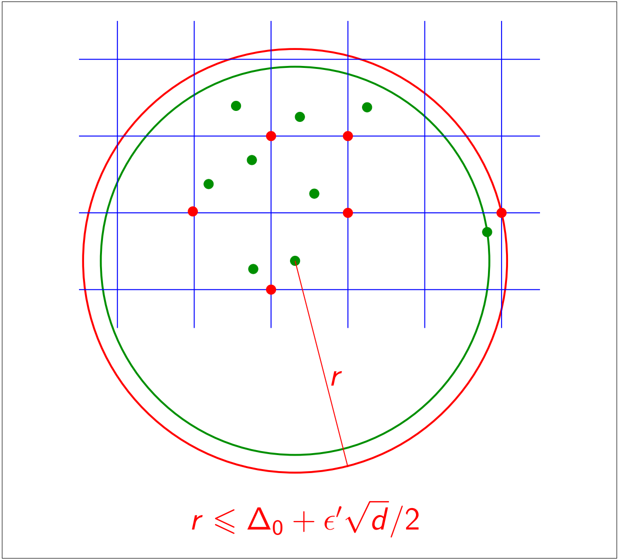
$$\begin{aligned}\Delta^* &\leqslant \left(1 + \epsilon\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16}\dots\right)\right)\Delta \\ &= (1+\epsilon)\Delta.\end{aligned}$$

- So Δ is a $(1+\epsilon)$-factor approximation of Δ^*.

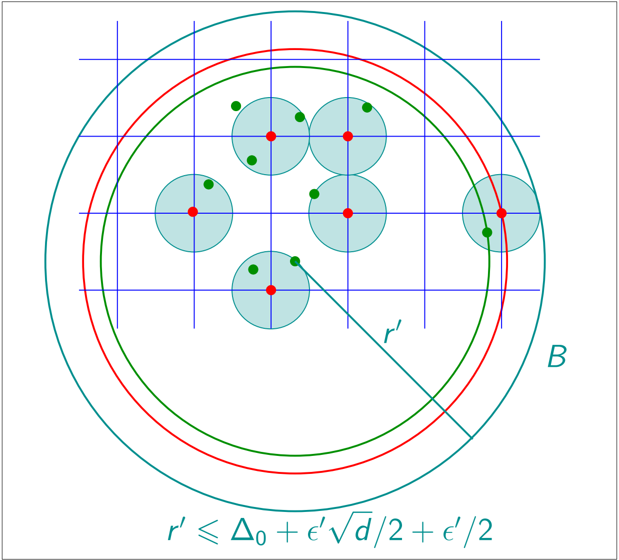
Cardinality of P'



Cardinality of P'



Cardinality of P'



Cardinality of P'

- All the points of P are in a sphere with radius Δ_0.
- So all the points of P' are in a sphere with radius $\Delta_0 + \epsilon'\sqrt{d}/2$.
- To each point $p \in P'$, we associate a ball $b(p)$ centered at p with radius $\epsilon'/2$.
- These balls are disjoint and contained in a ball B with radius $\Delta_0 + \epsilon'(\sqrt{d}/2 + 1/2) \leqslant 2\Delta_0$.

How Many Grid Points are There?

- The volume of a ball with radius r in dimension d is $C_d r^d$, where C_d depends only on d.
- So the number of balls $b(p), p \in P'$ is at most

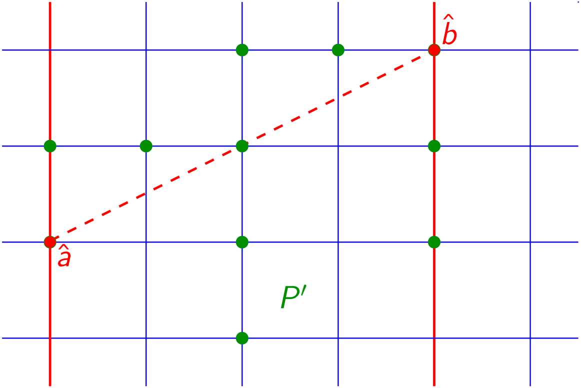
$$\frac{C_d(2\Delta_0)^d}{C_d(\epsilon'/2)^d} = \left(\frac{16\sqrt{d}}{\epsilon}\right)^d = O((1/\epsilon)^d).$$

- Each point $p \in P'$ is in exactly one ball $b(p)$.
- So $|P'| = O((1/\epsilon)^d)$.

Summary

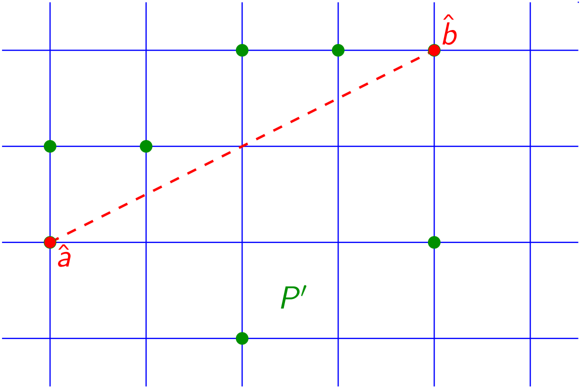
- Compute Δ_0 in $O(n)$ time.
- Round all the points to a grid with side length $\epsilon' = \epsilon\Delta_0/(4\sqrt{d})$.
  - It takes $O(n)$ time.
  - There are $O((1/\epsilon)^d)$ such points.
  - We denote by P' the set of rounded points.
- Compute the diameter Δ' of P' by brute force.
  - It takes $O((1/\epsilon)^{2d})$ time.
- $\Delta' - \epsilon'\sqrt{d}$ is a $(1+\epsilon)$-factor approximation of the diameter Δ^* of P.
- Running time: $O(n + (1/\epsilon)^{2d})$.

Grid Cleaning: Example in $\mathbb{R}^2$



- $\hat{a}$ is the lowest point on a vertical line
- $\hat{b}$ is the highest point on a vertical line

Grid Cleaning: Example in $\mathbb{R}^2$



- Idea: keep only the highest and the lowest point on each vertical line.
- Then run the brute-force algorithm.

Analysis in $\mathbb{R}^2$

- Only $O(1/\epsilon)$ points remain after grid cleaning.
- So the algorithm runs in $O(n + (1/\epsilon)^2)$ time.
- In higher dimension, same idea.
  - Consider all the points that coincide in their first $(d-1)$ coordinates.
  - Keep only highest and lowest.
  - Then only $O((1/\epsilon)^{d-1})$ remain from P'.
  - So rounding + grid cleaning yields a running time $O(n + (1/\epsilon)^{2d-2})$.

Projecting on Lines

- We measure angles in radian.
- That is, an angle is in $[0, 2\pi]$.

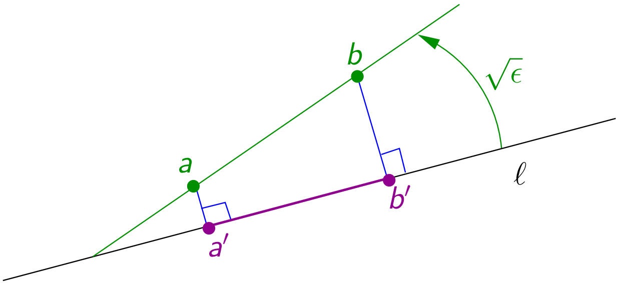
Property

For any α,

$$1 - \frac{\alpha^2}{2} \leqslant \cos\alpha \leqslant 1.$$

- Idea: We get a relative error ε by choosing α to be roughly $\sqrt{\varepsilon}$.

Projecting on Lines



$$|a'b'| \leqslant |ab| \leqslant (1+\varepsilon)|a'b'|$$

Projecting on Lines

- Assume that the angle between line ab and line ℓ is at most $\sqrt{\epsilon}$.
- a' (resp. b') is the orthogonal projection of a (resp. b) into ℓ.
- Then

$$\begin{aligned} |a'b'| &\leqslant |ab| \\ &\leqslant |a'b'| \times \frac{1}{1-\epsilon/2} \\ &= |a'b'| \times \left(1 + \frac{\epsilon}{2} + \frac{\epsilon^2}{4} + \frac{\epsilon^8}{8} + \dots\right). \end{aligned}$$

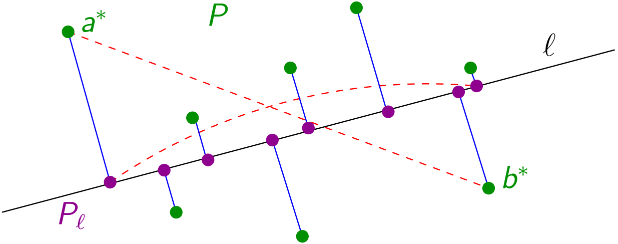
- Since we assume that $\epsilon < 1$, we obtain

$$|a'b'| \leqslant |ab| \leqslant |a'b'|(1+\epsilon).$$

- In other words, $|a'b'|$ is a $(1+\epsilon)$–factor approximation of $|ab|$.

Idea

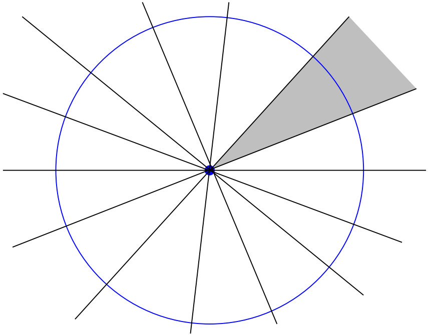
- P_ℓ obtained by projecting P onto a line ℓ.



- Compute the diameter of P_ℓ.
  - Can be done in $O(n)$ time: Find maximum and minimum along ℓ.

Idea

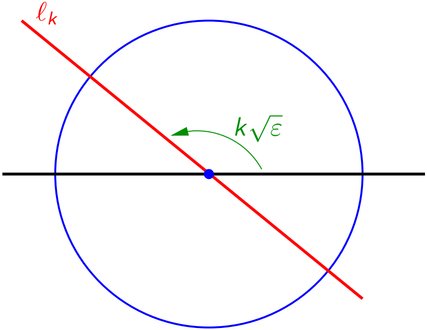
- If the angle between ℓ and a^*b^* is less than $\sqrt{\epsilon}$, then the diameter of P_ℓ is a $(1+\epsilon)$-factor approximation of Δ^*.
- How can we find a line that makes an angle $\sqrt{\epsilon}$ with ab?
  - Pick several lines.
  - In the plane: Pick direction $k\sqrt{\epsilon}$ for each $k \in [0, \pi/\sqrt{\epsilon}]$.



$O(\sqrt{\epsilon})$ cones
with angular diameter $\sqrt{\epsilon}$

Algorithm in $\mathbb{R}^2$

- For any integer $k \in [0, \pi/\sqrt{\epsilon}]$, we denote by ℓ_k a line that makes angle $k\sqrt{\epsilon}$ with horizontal.



- Project P onto ℓ_k, obtaining P_{ℓ_k}.
- The maximum of the diameter of P_{ℓ_k} over all k is a $(1+\epsilon)$-factor approximation of the diameter or P.

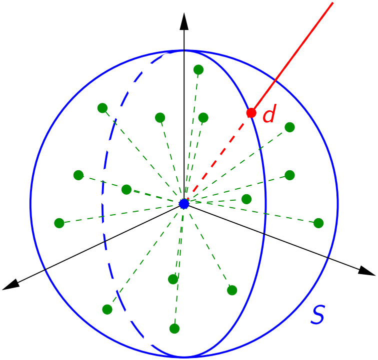
Algorithm in $\mathbb{R}^2$: Analysis

- Projecting onto a particular ℓ_k takes $O(n)$ time.
- Computing the diameter of P_{ℓ_k} takes $O(n)$ time.
- There are $O(1/\sqrt{\epsilon})$ such lines.
- Overall running time $O(n/\sqrt{\epsilon})$.

Algorithm in $\mathbb{R}^2$: Proof

- Let θ be the angle of $a^* b^*$ with horizontal.
- There exists k such that $k\sqrt{\epsilon} \leqslant \theta < (k+1)\sqrt{\epsilon}$.
- The angle between $a^* b^*$ and ℓ_k is at most $\sqrt{\epsilon}$.
- So the diameter of P_{ℓ_k} is at least $\Delta^*/(1+\epsilon)$.
- So the output of the algorithm is at least $\Delta^*/(1+\epsilon)$.
- On the other hand, the algorithm only looks at distances between two projected points, which are always smaller than Δ^*.

Generalization in $\mathbb{R}^d$



Generalization in $\mathbb{R}^d$

- Problem: Find a set of directions that approximates well the whole set of directions.
- Reformulation:
  - Let S be the unit sphere in $\mathbb{R}^d$.
  - Find $D \subset S$ with small cardinality such that $\forall x \in S$ there is a point $d \in D$ such that $|dx| \leqslant \sqrt{\epsilon}$.
  - A point $d \in D$ is associated with the line through the origin and d.
  - d handles a cone of direction with angular radius $\sqrt{\epsilon}$.
- Such a set D with cardinality $O((1/\epsilon)^{(d-1)/2})$ can be computed efficiently.
- So the algorithm runs in time $O(n(1/\epsilon)^{(d-1)/2})$.

Combining the two Techniques

- Running times:
  - Grid + cleaning: $O(n + (1/\epsilon)^{2d-2})$.
  - Projections: $O(n(1/\epsilon)^{(d-1)/2})$.
- Approach:
  - First round to P' and do grid cleaning.
  - We are left with $O((1/\epsilon)^{d-1})$ points.
  - Project on lines.
  - Overall running time: $O(n + (1/\epsilon)^{3(d-1)/2})$
  - Technical problem: the approximation factor is now about $1 + 2\epsilon$. (how to solve it?)